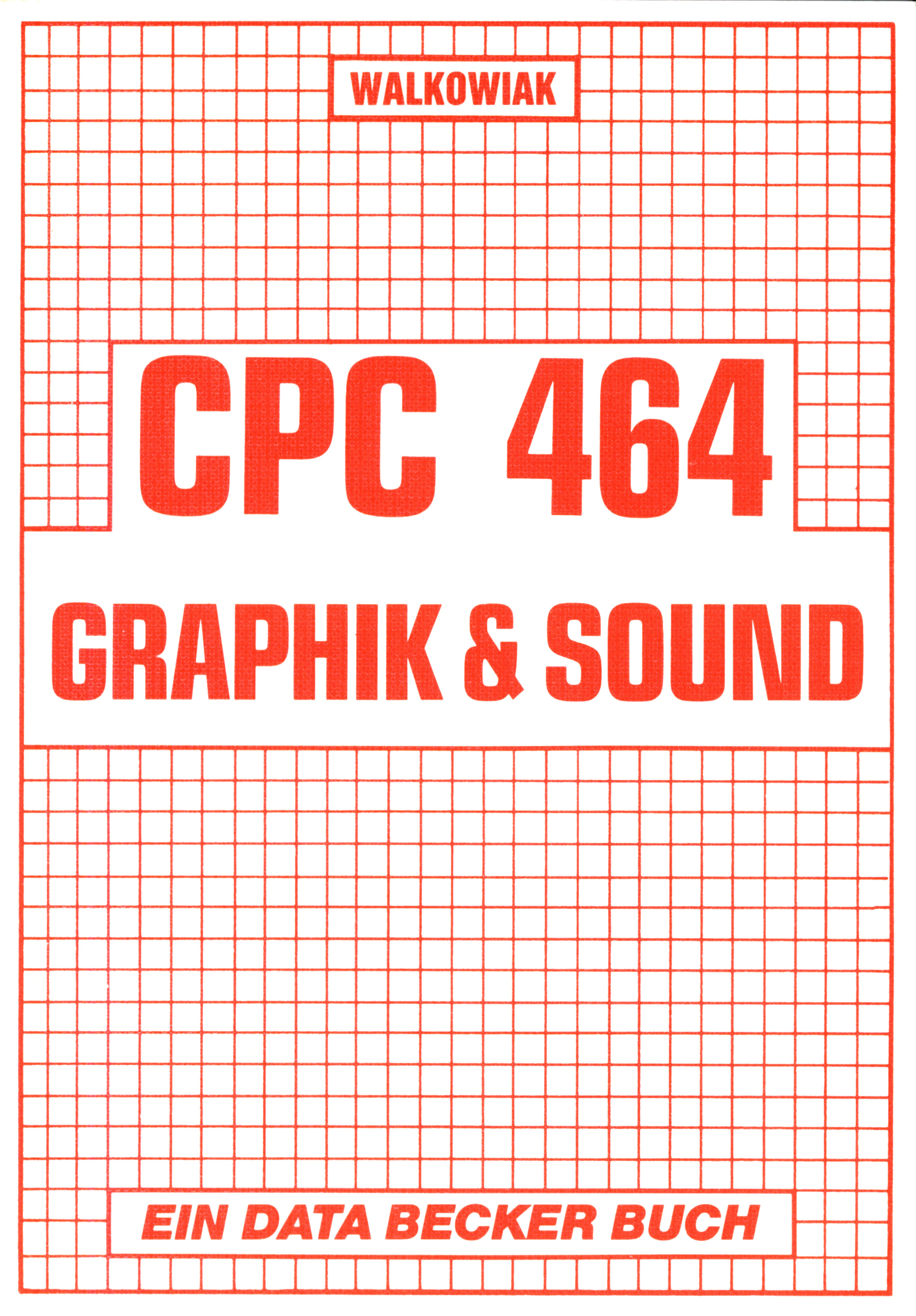
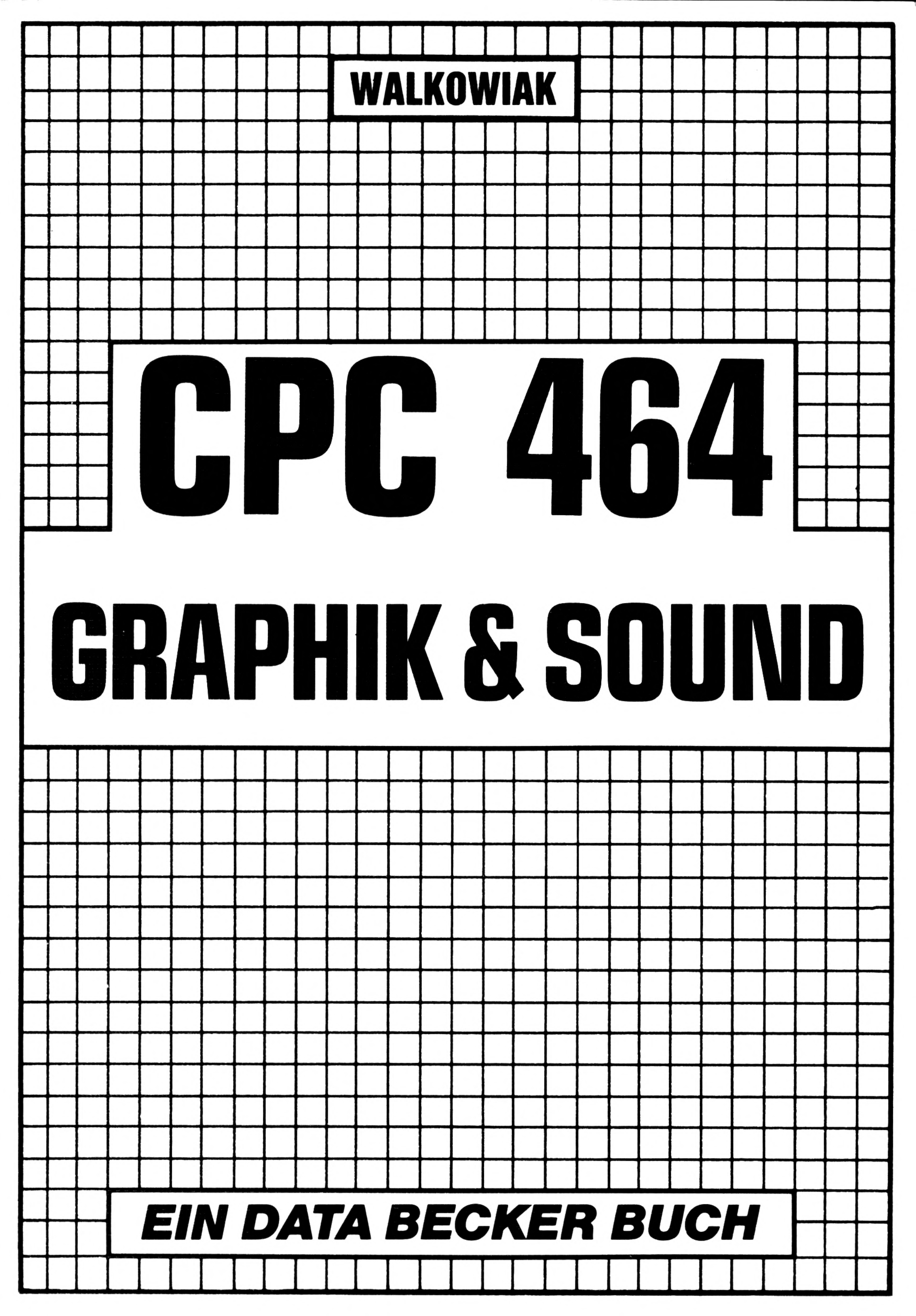

WALKOWIAK
CPC 464
GRAPHIK & SOUND
EIN DATA BECKER BUCH


WALKOWIAK
CPC 464
GRAPHIK & SOUND
EIN DATA BECKER BUCH

ISBN 3-89011-050-9

1. Auflage

Wichtiger Hinweis:
Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technischen Angaben und Programme in diesem Buch wurden von dem Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

INHALTSVERZEICHNIS

4. Kapitel - 2d Grafik

5. Kapitel - 3d Grafik

6. Kapitel - Sound

VORWORT

Grafik und Soundmöglichkeiten

- früher die Domäne besonders großer und teurer Spezialrechner - sind dank fortgeschrittener Technik heute auch dem kleinsten und preisgünstigsten Microcomputer zugänglich.

Doch was nützt dem stolzen Besitzer das Wissen, daß sich in seinem Heimcomputer ein Grafikprozessor wie beispielsweise der 6845-CRTC von Commodore oder ein mehrstimmiger Tongenerator wie der AY-3-8912 von General Instruments befinden, wenn er nicht weiß, wie er deren Eigenschaften dazu nutzen kann, um mathematische Funktionen grafisch darzustellen oder sein Keyboard in ein Orgelmanual zu verwandeln ?

Zwar gibt das mitgelieferte Handbuch Aufschluß über die Befehle, die zur Verfügung stehen, um diese IC's anzusprechen, doch die Algorithmen, also die Lösungsvorschriften für ein bestimmtes Problem, muß der Anwender selbst entwickeln oder entsprechender Literatur entnehmen.

Diesen mühsamen und zeitaufwendigen Prozeß will Ihnen dieses Buch ersparen, es will Ihnen ein unentbehrlicher Ratgeber und Lehrmeister sein und Ihnen helfen, alle Probleme der Grafik- und Soundprogrammierung zu meistern.

Im ersten Teil wird es Ihnen vom Punkt bis zum dreidimensionalem Bild anhand von zahlreichen Beispielprogrammen Techniken und Möglichkeiten der Grafikprogrammierung in Theorie und Praxis vorführen.

Gleiches gilt für die Programmierung von Tönen, Geräuschen und Melodien, denn diesen Themenkreis behandelt das letzte Kapitel des Ihnen nun vorliegenden Buches.

Nachdem Sie dieses Buch durchgearbeitet haben, werden Sie nicht nur über mehr Wissen verfügen, sondern es werden Ihnen auch eine Reihe von Programmen zur Verfügung stehen, für die Sie, im Falle, daß Sie sie käuflich auf Cassette oder Diskette erwerben, schon ein mehrfaches als den Kaufpreis dieses Buches ausgeben müßten.

So werden die Plotprogramme, welche Funktionen zwei- und dreidimensional darstellen, besonders für Schüler von großem Interesse sein.

Malerisch veranlagte Menschen werden ihre Freude an dem Grafikeditor haben und ihn zu ihrem neuen Zeichengerät machen.
Dabei wird er mit seiner Möglichkeit zur Erstellung von Unterprogrammen allen Programmierern, die Arcade-, Abenteuer- oder sonstige Spiele entwickeln, die Routinearbeiten zu einem reinen Vergnügen machen.

Der Geschäftsmann freut sich über die Plotprogrammme, welche auf Knopfdruck beliebige Statistiken als Balken- oder Kreisdiagramm darstellen.

Selbstverständlich darf auch die Kurzweil nicht zu kurz kommen, weshalb jeder Vergnügunssüchtige Tips & Tricks zur Programmierung von Actionspielen, wie auch ein komplettes Spiel in diesem Buch beschrieben findet.

Allerdings, Schießen und Raumschiffe fliegen ist nicht jedermanns Sache - wer sich lieber der leichten Muse hingibt, wird sich stattdessen mit der Miniorgel befassen.

Wie dem auch sei, ich nehme an, daß ich mit diesen Beispielen Ihr Interesse genügend geweckt habe, so daß ich diese Vorrede beenden und zum eigentlichen Thema des Buches kommen kann.

Sicher hingegen bin ich mir, daß Ihnen die Arbeit mit diesem Buch genau soviel Freude bereiten wird, wie mir das Schreiben und die Entwicklung der hier vorgestellten Programme.

Und damit auch alles seine Richtigkeit hat, haben wiederum die Lektoren Frau Alicia Clees und Herr Claus Wagner auf bewährte Weise zum Gelingen dieses Buches beigetragen, wofür ich ihnen an dieser Stelle recht herzlich danken möchte.

Recklinghausen,
März 1985 Jörg Walkowiak

1. KAPITEL
EINFÜHRUNG

KOMMUNIKATION IM WANDEL DER ZEIT

Würde ein Quizmaster in seiner Show den Kandidaten die Frage 'Welches ist die internationale Sprache ?' oder 'Welche Sprache wird von den meisten Menschen auf der Welt verstanden ?' stellen, so wird er vermutlich als Antwort 'Natürlich die englische !' erhalten.

Was, wenn die Antwort 'Die Bildsprache !' lauten würde ?

Denn wie war es wohl vor Urzeiten, als der Mensch noch auf allen vieren durch eine nur durch von Vulkanasche verschmutzte Umwelt spazierte und sich einem Stammesgenossen verständlich machen wollte ?

Begnügte er sich mit seinem Wortschatz, oder bediente er sich einiger Gesten und somit einer Zeichensprache ?
Ritzte er nicht vielleicht einige Striche in den Sand und schuf damit ein Bild, welches dem Anderen genau sagte, wo besonders gute Jagdgründe waren ?

Was gab es eher - handschriftlich gefertigte Dokumente oder Wandmalereien - einfache Strichzeichnungen mit Asche an eine Höhlenwand angebracht, aus denen die Historiker heute noch die Lebensgeschichten ganzer Sippen entnehmen können ?

Wie stand es mit der Kommunikation einige tausend Jahre nach dieser Zeit ?

Sehen wir uns altägyptische Hyroglyphen an, aus den Wandmalereien wurde eine Schrift, doch bestehen die einzelnen Buchstaben ebenfalls noch aus darstellenden Bildern.

Und wie ist es, wiederum einige tausend Jahre später, in unserer Zeit ?
Reicht uns unsere fortschrittliche Buchstabensprache, oder sprechen auch wir noch in Symbolen, in Bildern ?

Wie wenig Europäer kennen die Bedeutung des deutschen Wortes 'Frieden', wie viele hingegen wissen, was mit dem Bild einer weißen Taube gemeint ist ?
Wie viele Menschen auf der ganzen Welt kennen das Symbol des mit einem Pfeil durchbohrten Herzens, wie viele kennen das Pärchen von 'Liebe ist ...' ?

Bilder dienen somit heute noch immer der Verständigung der Menschen untereinander, haben sie doch den Vorteil, schnell und eindeutig zu informieren.

Dieser Tatsache ist man sich auch durchaus bewußt, wie sonst hätten Fotografie und Film zu ihrem Siegeszug antreten können?

Nicht umsonst gibt es die Illustrierten mit ihren Bildern im Vierfarbdruck, die Tages- und Wochenschau mit ihren bewegten Bildern, die Comics - wohl die einzige Literatur, die bereits die jüngsten unter uns interessiert - mit ihren ausdruckstarken, gefühlsbetonten Bildern, oder auch die technische Fachliteratur, versehen mit Plänen und Explosionszeichnungen.

Alle wissen es: 'Ein Bild sagt mehr als tausend Worte !', und so ist es kein Wunder, wenn die Bildsprache immer wieder auftaucht.

Sie hat zudem den Vorteil, immer den Erfordernissen entsprechend gestaltet werden zu können:

Urlaubsfotos müssen farbig und detailreich sein, sonst würden sie bei der lieben Verwandschaft wohl kaum einen Eindruck hinterlassen.

Ein Makler, der ein Haus verkaufen möchte, wird dieses dem potentiellen Käufer in seinem Büro zunächst auf einem Poster zeigen und diesem somit eine erste Entscheidungshilfe geben. Für den Ingenieur jedoch, der den Umbau vornehmen soll, wird dieses Bild völlig bedeutungslos sein, ihm wird eine Reihe von Zeichnungen mit den verschiedensten Ansichten lieber sein, und der Innenarchitekt wird auf keinen Fall auf den Grundriß verzichten wollen.

So begnügt sich der eine mit den Zeichnungen beziehungsweise Grafiken, welche nur die notwendigsten und für ihn wichtigen Informationen über ein Objekt enthalten, während der andere, nämlich der Käufer, auch Wert auf zusätzliche Informationen, wie beispielsweise die Farbgebung, legt, um sich 'ein Bild von den Verhältnissen zu machen'.

Aber Zeichnungen werden nicht nur dann gewählt, wenn es darum geht, bestimmte Eigenschaften realer Objekte darzustellen, sondern sie finden ihre Berechtigung auch dann, wenn es darum geht, Zahlen grafisch aufzubereiten.

Was dem Statistiker seine Gauß'sche Glockenkurve ist, das ist dem Schüler der Oberstufe die Darstellung der Ergebnisse seiner Kurvendiskussion, und das ist dem Geschäftsmann der Umsatzbericht des letzten Monats in Form von Balken- oder Tortengrafiken.

Aber auch in anderen Bereichen des täglichen Lebens wird man immer mehr mit künstlich erzeugten, auf das wesentliche reduzierten Bildern konfrontiert.

Da ist eine Fernsehstation besonders stolz auf ihr neues Emblem und meint, ihr Programm schon durch dieses teure Machwerk verbessert zu haben.

Da ist der Konstrukteur, der bislang seine Werkstücke in mühevoller Kleinarbeit unter Zuhilfenahme von Lineal, Stift und vielleicht Taschenrechner auf dem Reißbrett entwickelt hat, und nun in Windeseile sein Produkt mittels elektronischer Hilfsmittel auf dem Bildschirm zusammensetzt.

All diese Anwender profitieren von den Fortschritten der rasch voranschreitenden Computertechnik, die es nun auch kleineren Firmen und sogar Privatpersonen erlaubt, grafikfähige Syteme ihr eigen zu nennen.

Sie alle werden dazu beitragen, die Formen der Kommunikation um einen weiteren Aspekt zu bereichern, nämlich um die Computergrafik.

Denn es gibt schon seit längerem Wettbewerbe um die schönste Computergrafik; die Schlacht um zahlende Kunden, welche riesige Grafiksysteme zu Werbezwecken einsetzen, ist in vollem Gange, und auch der erste abendfüllende, vollsynthetische Spielfilm steht zur Zeit der Drucklegung dieses Buches vor der Tür.

COMPUTERGRAFIK

Computergrafik ist zur Zeit **das** Reizwort im Umgang mit Datenverarbeitungsanlagen.
Egal ob groß oder klein - womit an dieser Stelle sowohl der Rechner als auch sein Benutzer gemeint sein kann - keiner kommt mehr mit einem Computer aus, der **nur** rechnen oder Texte verarbeiten kann.
Und so wundert es auch niemanden, wenn die Anbieter der verschiedensten Systeme sich in einem andauernden Wettstreit laufend überbieten: ein größerer, schnellerer Speicher, eine höhere Auflösung oder noch mehr Farben lassen vergessen, daß gestern scheinbar noch eine benutzerfreundliche Tastatur einen guten Rechner ausmachte.

Aber was wird durch die neue Technik nicht alles möglich ?

Im professionellem Bereich der elektronischen Datenverarbeitung lassen sich zahlreiche Anwendungsbeispiele für Computergrafiken finden; in vielen Aufgabenbereichen ist der Einsatz eines Computers sogar erst in neuester Zeit aufgrund der gesteigerten grafischen Leistungen möglich geworden.

Versehen mit dem richtigen Programm ersetzt ein Großrechner ein ganzes Flugzeug - im Simulator können alle Aspekte des wirklichen Fliegens künstlich erzeugt und überzeugend dargestellt werden.

In der Tat gilt eine Ausbildung am Flugsimulator dem Realflug inzwischen als gleichgestellt, weshalb sie von den großen Fluggesellschaften, denen es auf diese Weise möglich ist, die Ausbildungskosten beträchtlich zu senken, auch als vollwertig anerkannt wird.

Galten die Computer im Bereich der Naturwissenschaften schon seit langem als unentbehrlich, weil niemand den enormen Berg der anfallenden Meßdaten in vertretbarer Zeit aufarbeiten und bislang auch kein Professor das gesamte Wissen der Menschheit im Kopf zugriffsbereit halten konnte, so sind sie heute, wo es um kompexe Zusammenhänge in der Plasmaphysik oder Genetik geht, unentbehrlich geworden.

Denn die menschlichen Sinne reichen schon lange nicht mehr, um dem Wissenschaftler bei seiner Arbeit eine Hilfe zu sein. Konnte er allerdings bislang sein Vorstellungsvermögen zu Hilfe nehmen, um sich ein Bild von den Ergebnissen seiner Berechnungen machen zu können, sieht er sich heute unüberwindbaren Problemen gegenüber, wenn es darum geht, komplizierte Erbstrukturen oder Moleküle zu vergleichen.
Hier ist das grafikfähige Terminal, auf dem sich die Moleküle in einer Simulation drehen und von allen Seiten vergrößert und verkleinert betrachtet werden können, mit Sicherheit eine unentbehrliche Arbeitshilfe.

Selbstverständlich müssen es nicht unbedingt Atomgruppen sein, die sich mathematisch erfaßt im Speicher des Computers befinden und deren Bildnisse dem Betrachter auf dem Videoschirm dargeboten werden, häufig sind es auch Autos und Häuser, ja sogar ganze Wohnblöcke.
Und befindet sich unter den Ausgabegeräten des Computers dann auch noch ein Plotter, dann kann sich der Architekt bei seiner Arbeit ganz und gar der künstlerischen Seite seiner Arbeit widmen.
Die Routinearbeiten, nämlich sämtliche Berechnungen durchzuführen und den gesamten, hoffentlich wohnlichen Komplex maßstabgetreu in sämtlichen Ansichten zeichnen, kann dann der Computer während der Pause oder in der Nacht durchführen.

Mit der Unterstützung seines Computers (daher CAD - Computer Aided Design) hat der Architekt qausi im Handumdrehen

Arbeiten erledigt, für die er früher Wochen benötigte.
In der Fertigunsindustrie können die so erarbeiteten Daten dann sogar in entsprechend ausgerüstete Drehbänke (CNC-Maschinen) eingespeist werden, so daß der Mensch seine bildhaften Vorstellungen eines Objektes vollautomatisch realisieren kann.
Übrigens wird der gesamte Prozeß dann als CAM, als Computer Aided Manufacturing, bezeichnet.

Aber nicht nur in Forschung und Industrie können diese Supercomputer eingesetzt werden, sondern dank ihrer hervorragenden Grafikeigenschaften werden sie auch immer mehr in künstlerischen Bereichen eingesetzt.

So entstehen in Amerika und Japan Computergrafik-Studios, die mit ihren Arbeiten zeigen, daß selbst die ausgefallensten Objekte überzeugend dargestellt werden können, wobei die Bildqualität in Hinblick auf Schärfe und Farbtreue so hervorragend ist, daß bereits der erste Science Fiction Film, welcher unter dem Titel 'The Works' am New Yorker Institute of Technology entsteht, kurz vor seiner Uraufführung steht.

Wurden bei 'TRON' in den Disney-Studios noch Realfilm und Computergrafik zusammenkopiert, so verwenden die New Yorker Produzenten nur noch synthtische Sequenzen.
Es scheint also, daß die Zeiten der Blue Screen Technik demnach bereits vorüber sind.
Vermutlich werden es in einigen Jahren überbezahlte Schauspieler sein, die sich vor dem 'Kollegen Computer' fürchten.

Gerade dieses Beispiel zeigt, wie perfekt heutige Computergrafik sein kann, feinste Differenzierungen des Computerbildes sind möglich.
Dies zeigt auch der Einsatz von Grafiksystemen in der

Medizin - feinste Gewebeveränderungen macht der Tomograph dem Arzt auf einem Bildschirm sichtbar und dieser kann dadurch unzähligen Menschen helfen.

All diese Anwendungen wären undenkbar, wenn es nicht möglich geworden wäre, Bilder mit all ihrer Vielfalt an Farben und Formen unter Zuhilfenahme einer entsprechenden technischen Ausrüstung zu erzeugen.
Obige Beispiele haben gezeigt, daß die Grafikfähigkeiten eines Systems sich als unabdingbare Voraussetzung zu den verschiedensten Computeranwendungen zeigen.
Und so wird denn auch ständig weitergeforscht und entwickelt, es werden neue Wege gesucht und probiert; der eine Hersteller hat das System des anderen unauffällig kopiert, dabei aber auch perfektioniert, ein anderer präsentiert dafür etwas völlig neues.

Glücklicherweise profitieren davon nicht nur die Finanzgiganten, sondern es kommt dabei auch etwas für den kleinen Mann heraus.

Denn auch dem kleinen Anwender, insbesondere aber den kleinsten, geht es um Punkte und Linien, um Sprites und Shapes, um Farben und um Formen.
Computergrafik ist somit nicht nur ein Schlagwort der Industrie, sondern auch Schlüsselwort zu den Herzen zahlloser Kinder und Jugendlicher, denen viel daran liegt, das neueste Spielhallenspiel in den eigenen vier Wänden zu haben.

Kein Heimcomputer ließe sich heute verkaufen, wenn er nicht in der Lage wäre, farbige Weltraumlandschaften auf dem heimischen Bildschirm zu erzeugen, und so ist es kein Wunder, wenn auch die Herstellerfirmen, die den Heimbereich für sich erschlossen haben, ihren Computern einige grundlegende Fähigkeiten mitgeben, die auch auf diesen Microcomputern durchaus respektable Ergebnisse erzielen lassen.

2. KAPITEL
GRUNDLAGEN

Bevor wir uns richtig mit der Computergrafik befassen und uns um Probleme wie Drehungen, Spiegelungen oder auch Skalierungen kümmern, sollen anhand kurzer Beispiele noch einmal die grundsätzlichen Grafikbefehle des Schneider CPC behandelt werden.
Sicherlich werden Ihnen die folgenden Zeilen zunächst wie eine Wiederholung aus dem Handbuch vorkommen, dennoch sollten Sie dieses Kapitel nicht überblättern, da Sie dieser Einführung sicherlich noch einige Details entnehmen können, welche sich als grundlegend für die späteren Arbeiten zeigen.

Die Listings

in diesem Buch wurden in einer Form abgedruckt, wie sie auch auf dem Bildschirm Ihres Computers erscheinen, wenn die Standardbetriebsart eingeschaltet ist.
Dies ermöglicht Ihnen eine schnelle Kontrolle der Programme, wenn sie einmal nicht so laufen sollten wie hier beschrieben.

Einige der vorgestellten Routinen tauchen in den verschiedensten Programmen wieder auf.
Aus diesem Grunde wurden sie zum Teil als Unterprogramme ausgestaltet, weshalb Sie darauf verzichten sollten, die RENUMBER-Funktion des Locomotive-Basic einzusetzen.
Denn nur dann können Sie diese Programmteile mit MERGE in der gewünschten Art und Weise zusammenfügen und sich somit einige Tipparbeit ersparen.

Die Technik

die sich in unserem CPC 464 befindet, ermöglicht es uns zwar nicht, ebenfalls real erscheinde Filmbilder zu erzeugen, bereitet uns aber doch den Weg zu einer Reihe von effekt- und auch sinnvollen Anwendungen.

Wie sich bereits dem vorangegangenen Text entnehmen ließ, erfordern ansprechende Grafiken ein möglichst hohes Auflösungsvermögen wie auch eine Vielzahl von Farbschattierungen.
Selbstverständlich ist unser Computer der 1000-DM-Klasse in Bezug auf seine Leistung noch weit von den Fähigkeiten einer mehrere Millionen Mark teuren Cray, welche eine Million einzelner Punkte auf einem Grafikterminal darstellen kann, entfernt, doch verschaffen die immerhin 128.000 einzeln ansprechbaren Punkte, die im Fachjargon übrigens Pixel heißen, dem Schneider CPC in einer Vergleichsliste der gängigsten Microcomputer, wie Apple, Atari, Sinclair und Commodore, den ersten Platz.

Es ist für jedermann einzusehen, daß der Computer sich in seinem Speicher merken muß, ob ein einzelnes Element des Bildes nun leuchten oder dunkel sein soll; ist jedem Punkt auch noch eine bestimmte Farbe zugeordnet, so wird sich dieser Speicherbedarf noch vergrößern.

Aus diesem Grunde sind die Hersteller gezwungen, Kompromisse zu schließen, die dann folgendermaßen aussehen können:

1. hohe Auflösung, wenige Farben
2. niedrige Auflösung, Farbenvielfalt
3. hohe Auflösung, differenzierte Farbdarstellung

Punkt eins und zwei kamen im Schneider-Computer zur Anwendung; Punkt drei wäre technich ebenfalls in einem Rechner mit Heimcomputergröße machbar, doch würden die

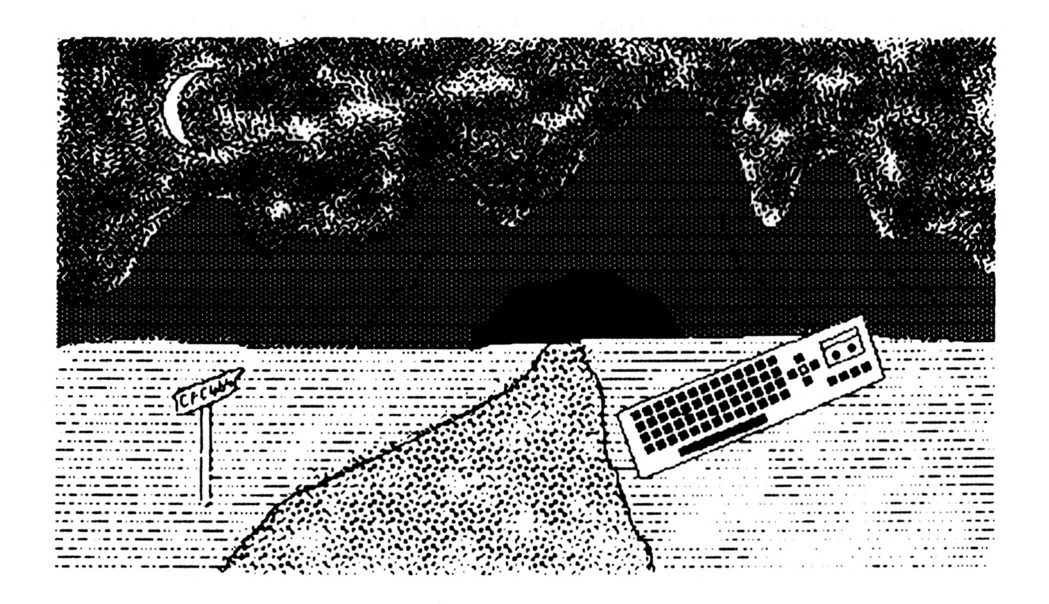
erforderlichen Speicherbausteine und der notwendige bessere Monitor den Preis für ein solches Gerrät dermaßen in die Höhe treiben, daß wir uns diese Computer dann nicht mehr leisten könnten.
Zumindest heute noch nicht, denn waren die Rechner im Jahre 1977 noch mit bestenfalls 16k-Chips ausgerüstet und hatten sie eine grafische Auflösung von vielleicht 127 x 48 Punkten (TRS-80), so finden wir heute 64k-Chips, also Speicherbausteine mit vierfachem Fassungsvermögen, und Auflösungen von 600 x 200 Punkten (Schneider CPC).
Pressemitteilungen einiger Halbleiterhersteller läßt sich aber entnehmen, daß bereits Prototypen von 1MB-Chips existieren - wenn diese in Serienproduktion gehen und nicht mehr mehrere hundert Mark pro Stück kosten, werden wir wohl eine weitere Leistungssteigerung der Heimcomputer erleben.

Das Zeichengerät

Wohl jeder von uns hat bereits mit den unterschiedlichsten Mitteln versucht, sich künstlerisch zu betätigen, sei es nun im Kindergarten mit Wand- oder Wasserfarbe, oder in der Schule mit Bleistift und Tuschefüller, oder aber auch zu Hause mit Glas- beziehungsweise Ölfarbe.

Alle Techniken hatten eines gemeinsam: So wie der Maler zunächst erst seine Staffelei aufbaut, so haben auch wir uns erst die Arbeitsmaterialien besorgt, haben den Bildträger gewählt und uns unsere Lieblingsfarben aus dem erhältlichen Angebot ausgesucht, dann erst, nach einer Phase der Besinnung, haben wir mit der eigentlichen Arbeit begonnen.

Auch wenn wir nun unseren Schneider CPC einsetzen, so werden wir wiederum zunächst den Untergrund, das Papier, wählen, und uns Gedanken über die zu verwendenden Farben machen.



Nun sollten Sie ihren CPC einschalten !

PAPER

Selbst die nicht englischsprechenden unter uns wissen sofort, was gemeint ist, denn mittels dieser Anweisung wählen wir eine von sechsundzwanzig möglichen Farben für den Hintergrund aus:

```
10 MODE 0:PAPER 0:PEN 1
20 FOR FARBE=0 TO 26
30 INK 0,FARBE
40 INK 1,FARBE+1
50 LOCATE 1,1:PRINT "FARBE: ";FARBE
60 FOR I=1 TO 1000:NEXT I
70 NEXT FARBE
```

Dieses kurze Programm präsentiert uns die Palette aller siebenundzwanzig verfügbaren Farben:

0	schwarz	1	blau
2	hellblau	3	rot
4	magenta	5	hellviolett
6	hellrot	7	purpur
8	helles magenta	9	grün
10	blaugrün	11	himmelblau
12	gelb	13	weiß
14	pastellblau	15	orange
16	rosa	17	pastellmagenta
18	hellgrün	19	seegrün
20	helles blaugrün	21	limonengrün
22	pastellgrün	23	pastellblaugrün
24	hellgelb	25	pastellgelb
26	leuchtendweiß		

Ergänzen Sie obiges Programm um nachstehende Zeilen, und Sie können sich alle Farben nach Wunsch noch einmal in Ruhe ansehen:

```
90 INPUT "HINTERGRUNDFARBE ";FARBE
100 INK 0,FARBE:INK 1,FARBE+1
110 GOTO 90
```

BORDER

Es ist Ihnen sicherlich aufgefallen, daß um die eigentliche Zeichenfläche herum immer noch ein Rand bestehen bleibt, den Sie nicht beschreiben können.
Sollten Sie diesen auffälligen Rahmen nicht mögen, können Sie ihn mit dem Befehl BORDER f, wobei f einer der möglichen Farbnummern von 0 bis 26 entspricht, genau wie den Hintergrund einfärben.

PEN, INK

Pen ist der von Ihnen gewählte Schreibstift. Falls Sie die Zusammenhänge zwischen Paper, Pen und Ink noch nicht mittels des Schneider-Handbuches erarbeitet haben, dann stellen Sie sich unter einem Pen am besten einen Kugelschreiber in einem Etui vor.
Genau so, wie sich in einem Etui mehrere Stifte befinden können, so erlaubt Ihnen der CPC auch den Gebrauch mehrerer Stifte, wobei deren maximale Anzahl von der gerade gewählten Betriebsart abhängig ist.

Jeder dieser Stifte enthält dabei zunächst eine für ihn typische Mine, so befindet sich in Stift 0 eine Mine der Farbe 1 (blau) und in Stift 1 eine von gelber (24) Farbe.
Die Standardwerte der folgenden Pens sind dann von der gerade eingestellten Betriebsart abhängig.
So erlaubt Mode 2 nur die Verwendung zweier Farben, weshalb alle weiteren Stifte alternierend die gleichen Minen enthalten.

In der Betriebsart 1 kann der Schneider vier Farben gleichzeitig darstellen, so daß sich die Gruppe blau, gelb, blaugrün und rot ständig wiederholt.

Beachtenswert sind die Stifte 14 und 15 der Betriebsart 0, denn diese schreiben mit zwei ständig wechselnden Farben.

Wem diese Einstellung nicht zusagt, der kann, ebenso wie er auch die Schreibmine seines Kugelschreibers wechseln kann, mittels der Anweisung INK den erlaubten Stiften andere Farben zuordnen, wobei nun die Zahlenwerte obiger Tabelle gelten. Werden gleich zwei Farben einem Stift zugeordnet, so ist das Ergebnis wieder ein verwirrendes Farbspiel.
Diese Färbetechnik gilt dabei auch für Hintergrund und Rahmen, denn auch diesen ist immer ein Schreibstift zugeordnet.

Ein kurzes Beispiel soll den Gebrauch von PEN und INK demonstrieren:

```
10 REM FARBBALKEN
20 MODE 0:PAPER 0
30 zeile$=STRING$(40,143)
40 FOR x=1 TO 13
50 PEN x
60 PRINT zeile$
70 NEXT x
```

```
80 FOR farbe=0 TO 26
90 farbe2=farbe
100 FOR x=1 TO 13
110 INK x,farbe2
120 farbe2=farbe2+1
130 IF farbe2>26 THEN farbe2=0
140 FOR i=1 TO 100:NEXT
150 NEXT x
160 NEXT farbe
170 GOTO 70
```

Die Betriebsart

werden Sie jeweils nach den Erfordernissen auswählen. Charakteristisch sind dafür:

1. die Anzahl der Farben
2. die Größe der Buchstaben
3. die Auflösung.

So stehen Ihnen in MODE 0 sechzehn der siebenundzwanzig Farben zur gleichzeitigen Anwendung zur Verfügung, da Sie in dieser Betriebsart 16 Stifte, also PEN's, benutzen können.
Die einzelnen Buchstaben sind riesig, weshalb auch nur noch zwanzig von ihnen in eine Bildschirmzeile passen.
Ebenso riesig sind die kleinsten, einzeln darstellbaren Punkte, denn die Auflösung beträgt nur 160 mal 200 Punkte.
Gerade richtig dürfte diese Einstellung für die Ausgabe von Titelbildern sein; programmieren werden Sie wohl kaum in dieser Betriebsart.

Mit MODE 1 wählen Sie die Standardeinstellung: vierzig Buchstaben in 24 Zeilen, vier Stifte und eine Auflösung von 320 mal 200 Punkten.

Diese Betriebsart wird nicht nur einfach deshalb als Normal bezeichnet, weil der CPC sich nach dem Einschalten in Mode 1 befindet, sondern auch deshalb, weil es sich um die typischen Bildschirmdaten eines Heimcomputers handelt.
Denn die bislang üblichen Geräte wurden keinesfalls zusammen mit einem Monitor geliefert, sondern mußten sich mit dem heimischen Fernsehapparat begnügen, und dieser war und ist keinesfalls in der Lage, mehr als die in dieser Betriebsart ansprechbaren Punkte befriedigend darzustellen.

Deshalb findet sich auf dem Gehäuse des CPC 464 wohl auch die Bezeichnung Colour Personal Computer, denn die Auflösung von 640 mal 200 Punkten in Mode 2 übertrifft die Werte des IBM PC's !
Und auch die achtzig Zeichen innerhalb einer Zeile finden sich sonst nur bei Geräten für den professionellen Einsatz.

Nun sind die Anzahl der Farben wie auch die Größe der Buchstaben recht gut abzuschätzende Kriterien.
Die Auflösung läßt sich allerdings nicht so herrlich intuitiv beurteilen.
Dies läßt sich am deutlichsten unter Zuhilfenahme eines kurzen Beispielprogrammes zeigen, welches in der Art eines Testbildes, wie es auch bei Fernsehstationen verwendet wird, das Auflösungsvermögen der verschiedenen Betriebsarten vor Augen führt.

```
10 REM Demonstration: Aufloesung
20 REM ------------------------
30 PAPER 0:INK 0,12:BORDER 12:PEN 1:INK
1,0
40 y1=100:y2=300
50 CLS:INPUT"Welche Betriebsart  (-1,0,1
,2)";modus:IF (modus>2 OR modus<-1) THEN
GOTO 50 ELSE MODE modus
```

```
60 IF modus=-1 THEN MODE 1:END
70 LOCATE 1,1:PRINT"Testbild: Aufloesung
"
80 PLOT 100,100:DRAW 510,100:DRAW 510,30
0:DRAW 100,300:DRAW 100,100
90 TAG
100 FOR x=80 TO 290 STEP 21
110 GOSUB 390
120 NEXT x
130 MOVE 76,98:PRINT CHR$(240);"20";
140 FOR x=290 TO 400 STEP 11
150 GOSUB 390
160 NEXT
170 MOVE 296,98:PRINT CHR$(240);"10";
180 FOR x=400 TO 450 STEP 5
190 GOSUB 390
200 NEXT
210 MOVE 396,98:PRINT CHR$(240);"5";
220 FOR x=450 TO 490 STEP 4
230 GOSUB 390
240 NEXT
250 IF modus>0 THEN MOVE 446,98:PRINT CH
R$(240);"3";
260 FOR x=490 TO 520 STEP 3
270 GOSUB 390
280 NEXT
290 IF modus>0 THEN MOVE 476,98:PRINT CH
R$(240);"2";
300 FOR x=520 TO 540 STEP 2
310 GOSUB 390
320 NEXT
330 IF modus>1 THEN MOVE 496,98:PRINT CH
R$(240);"1";
340 FOR x=540 TO 550
350 GOSUB 390
360 NEXT
370 TAGOFF
```

```
380 IF INKEY$="" THEN 350 ELSE 50
390 PLOT x,y1:DRAW x,y2:RETURN
```

Nach dem Start mit RUN fragt Sie das Programm nach der Betriebsart. Wählen Sie durch Eingabe einer zwei bitte eine Auflösung von 640 mal 200 Punkten.

Nachdem Sie dann <RETURN> gedrückt haben, erscheinen innerhalb eines Fensters in der Schirmmitte dann einige Linien, die in einem genau definierten Abstand zueinander stehen.
Dieser beträgt zwischen den ersten zehn Zeilen von links genau die zwanzigfache Stärke der Striche, danach befinden sich zwischen den jeweils zehn einzelnen Teilstrichen einer Gruppe Leerräume von der zehn-, fünf-, drei-, zwei- und einfachen Breite dieser Markierungen.

Lassen sich im Bereich der fünfer-Abstände die einzelnen Linien noch einwandfrei erkennen, so haben wir im dreier-Block bereits große Schwierigkeiten, die Anzahl der Linien festzustellen.
Innerhalb des Zweierblocks ist ein Durchzählen auf einem Farbmonitor bereits unmöglich, und wenn auf jede gezeichnete Linie eine Leerspalte folgt, zeigt sich das Feld bereits als solider Block, durch welchen sich ein seltsames Muster zieht.

Drücken wir nun eine beliebige Taste und wählen anschließend Mode 1. Bereits in der Dreiergruppe fällt es uns schwer, noch einzelne Linien zu erkennen, und in der Betriebsart 0 erscheint dieser Ausschnitt des Testbildes bereits als ausgefüllte Fläche.

Dieses Beispiel macht wohl recht deutlich, was unter dem Begriff der Auflösung zu verstehen ist.

MOIRE

An dieser Stelle sind wir gar nicht einmal mehr so weit von unseren ersten ansprechenden Computergrafiken entfernt.

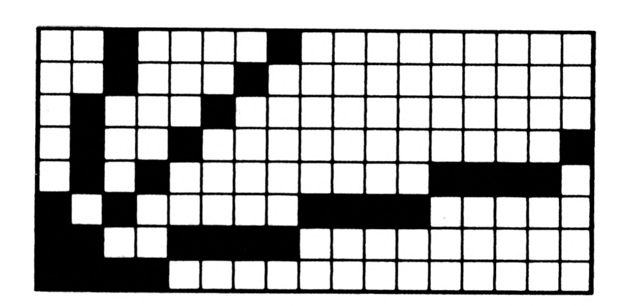
Denn das bei unserem Testbild in Mode 2 aufgetretene Flimmern ist häufig Grundlage zu Computergrafiken, die einfach nur ästhetisch schöne Bilder sein sollen und nicht irgendwelchen anderen Zwecken als der Kunst dienen.

Bezeichnet wird dieser Effekt mit 'Moire' - nach einem Gewebe, welches durch seine spezielle Bearbeitung eine eigenartig schillernde Oberfäche aufweist.
Sie alle kennen diese Erscheinung auch von Ihrem Farbfernsehgerät her, wenn sich über eine feingemusterte Fläche einige Farbschlieren ziehen, die dort nichts zu suchen haben.
Dort, wie auch hier, findet sich die Bildröhre als der Verursacher für diesen Effekt, denn sie ist nicht mehr in der Lage, die feinen Details einwandfrei darzustellen:

```
10 REM MOIRE (MONITOR)
20 ------------------
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
40 FOR x=1 TO 640 STEP 2
50 PLOT x,1:DRAW x,400
60 NEXT
```

Und, wie wir längst wissen, sind auch dem Computer selbst Grenzen gesetzt. Zum einen ist selbst der kleinste darstellbare Punkt immer noch ein Rechteck, wie sich durch die Eingabe von MODE 0:PLOT 1,1 beweisen läßt.
Zum anderen hat dies zur Folge, daß eine Gerade auf dem Bildschirm niemals eine Gerade, sondern eher eine Treppe

ist, wie folgende Skizze deutlich macht:



Somit wird auch ein Strahlenbüschel kaum als solches erscheinen, sondern führt zu weitaus interessanteren Ergebnissen:

```
10 REM MOIRE 2
20 DEFINT A-Z
30 MODE 2
40 FOR X=0 TO 640 STEP 4
50 PLOT 0,0:DRAW X,400
60 PLOT 640,400:DRAW X,0
70 NEXT
80 IF INKEY$="" THEN 80 ELSE END
```

```
10 REM MOIRE 3
20 DEFINT A-Z
30 MODE 2
40 FOR Y=400 TO 0 STEP -3
50 PLOT 0,0:DRAW 640,Y
60 PLOT 640,400:DRAW 0,Y
70 NEXT
80 IF INKEY$="" THEN 80 ELSE END
```

Kleine Änderungen bedeuten wiederum andere Ergebnisse, so daß Sie probeweise Mode 2 durch Mode 1 ersetzen oder auch die Schrittweiten hinter Step abändern sollten.

Um solche Grafiken auf einem beliebigen Computersystem erzeugen zu können, müssen Sie sich nur über dessen Koordinatensystem und über dessen Befehlsfolge zur Erzeugung einer Linie im klaren sein.

Nun, bei unserem Schneider haben wir den Befehl PLOT x,y um einen Punkt zu setzen, und DRAW x,y, um von dem zuletzt gesetzten Pixel eine Linie bis zu dem Punkt x,y zu ziehen, wobei die Werte für x und y nach den üblichen, in der Schule trainierten Konventionen eines kartesischen Koordinatensystemes vergeben werden.
Allerdings gilt es dabei zu beachten, daß uns nach dem Einschalten des Gerätes nur der erste Quadrant zur Verfügung steht, x und y somit nur positive Werte annehmen können.

Mit diesem Wissen ausgestattet, können wir, jeweils durch kleine Änderungen an unserem Programm, zahllose voneinander verschiedene Grafiken erzeugen.
Denn die einzelnen Linien müssen ja nicht, wie bei obigen Beispielen, von den Eckpunkten des Bildschirmes ausgehen, sondern die Strahlen können an jedem beliebigen Punkt beginnen. Weiterhin kann es auch mehrere Ursprungspunkte geben, die Strahlen können sich schneiden, oder die Schrittweite kann sich von Linie zu Linie ändern; unterschiedliche Farben innerhalb eines Bildes können ebenfalls zu dessen Gestaltung beitragen.

Wie es gemacht wird, zeigt das folgende Programm, welches trotz seiner Kürze betrachtenswerte Grafiken erzeugt:

```
10 REM COMPUTERGRAFIK 1
20 REM 1 URSPRUNGSPUNKT
30 REM ----------------
40 DEFINT a-z
50 PAPER 0:PEN 1:INK 0,12:INK 1,0:BORDER
12
60 MODE 2
80 x=RND(1)*640:y=RND(1)*400
90 schrittweite=RND(1)*10+2
100 FOR x1=1 TO 640 STEP schrittweite
110 PLOT x,y:DRAW x1,400
120 PLOT x,y:DRAW x1,1
130 NEXT x1
140 FOR y1=400 TO 1 STEP -(schrittweite/
2)
150 PLOT x,y:DRAW 640,y1
160 PLOT x,y:DRAW 1,y1
170 NEXT y1
190 FOR i=1 TO 10000:NEXT
200 GOTO 60
```

Um nun Grafiken mit mehreren Ursprungspunkten erzeugen zu können, müssen Sie das Programm ein wenig abändern.

Ergänzen Sie eine weitere Schleife und legen Sie die Zahl der gewünschten Punkte mit dem Schleifenendwert in Zeile 70 fest.
Möglicherweise empfiehlt es sich dann aber, die Mindestschrittweite zu erhöhen, so daß Ihr Kunstwerk nicht schwarz in schwarz verläuft.

```
70 FOR punkt=1 TO 2
180 NEXT punkt
200 CLS:GOTO 70
```

So weit - so gut !

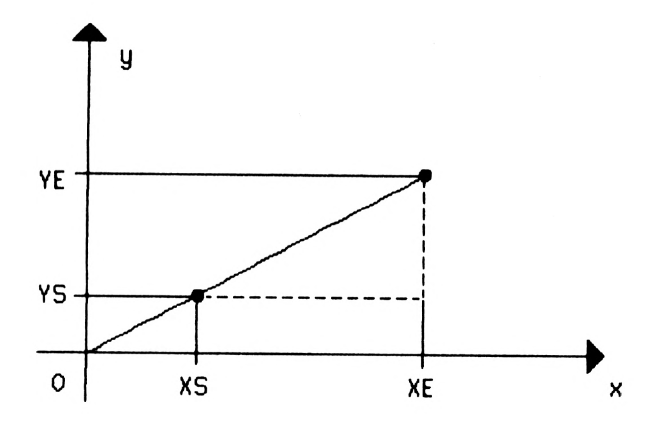
Wer das Buch bis zu dieser Stelle gelesen oder gar durchgearbeitet hat, der weiß, wie er sich der verschiedenen Farben bedienen kann, wie Punkte gesetzt und Linien gezeichnet werden.

Doch was tun, wenn die Darstellung einen Kreis erfordert ?
Oder wenn eine präzise Zeichnung mehrere deutlich voneinander unterscheidbare Linientypen enthalten soll, wir bei hoher Auflösung aber nur eine Zeichenfarbe haben ?
Oder wenn von vornherein feststeht, daß die Grafik auf einem Drucker oder Plotter ausgegeben werden soll, der nicht über mehrere Fabstifte verfügt ?

Unser CPC hilft uns hier nicht weiter, wohl aber die Mathematik !
Und so wird es sich nicht vermeiden lassen, daß von nun an neben der spielerischen Beschäftigung mit der Computergrafik auch mathemathische Formeln und Gesetze im Vordergrund des Buches stehen.

Linien

lassen sich schnell und genau mit der Anweisung DRAW x,y zeichnen.
Doch machen es die eben aufgeführten Gründe wünschenswert, einen eigenen Algorithmus zu entwickeln, welcher in der Lage sein soll, unterbrochene Linien auf den Schirm zu bringen.
Um das Problem in den Griff zu bekommen, sehen wir uns zunächst die folgende Zeichnung an:



Irgendwo innerhalb des erlaubten Bereiches unserer Zeichenfläche befinden sich zwei Punkte, die wir miteinander verbinden wollen. Alles was wir von ihnen wissen, sind ihre x- und y-Koordinaten.

Ihren Abstand voneinander können wir zunächst nur für jede Achse getrennt angeben:

```
dx = xe - xs
dy = ye - ys
```

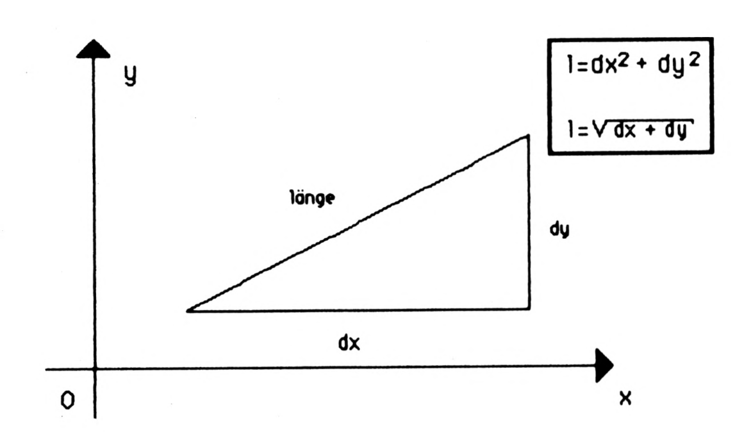
Nicht ablesen läßt sich hingegen die kürzeste Verbindung der zwei Punkte, die Gerade, die wir zeichnen wollen.

Erinnern wir uns daher an einen alten Gelehrten namens Pythagoras und machen uns dessen Lehrsatz, daß die Summe der Quadrate in einem rechtwinkligem Dreieck gleich dem Quadrat über der Hypotenuse sei, zu eigen.

Auf unser Beispiel angewandt, zeigt sich dann, weil die gesuchte Strecke eben dieser Hypotenuse entspricht, daß wir ihre Länge mit

```
laenge = SQR(dx * dx + dy * dy)
```

ansetzen müssen. Somit ist die Anzahl der zu setzenden Punkte bekannt, und wir sind einen großen Schritt weiter.



Um nun auch noch die genauen Koordinaten eines jeden Punktes auf dieser Linie festlegen zu können, teilen wir die Länge dieser Strecke durch die im ersten Schritt festgestellten Differenzbeträge, und wissen dann genau, welchen Anteil wir zu den Koordinaten unseres Ausgangspunktes hinzuzählen müssen:

Anteil x = dx / länge
Anteil y = dy / länge

==> x = xs + SCHRITT * dx
y = yb + SCHRITT * dy

SCHRITT ist dabei das gerade zu zeichnende Teilstück (eben genau ein einziger Punkt) der Verbindungslinie, und hängt deshalb von der Schrittgröße der Schleifenvariablen ab (die wiederum davon abhängt, wie dicht die Linie gezeichnet werden soll).

Als Ergebnis all dieser Überlegungen können wir nun folgendes Programm formulieren:

```
10 REM DRAW LINE ROUTINE
20 MODE 2:PEN 1:INK 1,24:BORDER 0:INK 0,
0
30 DEFINT A-Z
40 INPUT "von welchem x ";xs
50 INPUT "und welchem y ";ys
60 INPUT "nach welchem x ";xe
70 INPUT "und welchem y ";ye
80 INPUT "jeden wievielten Punkt setzen
";sw
90 dx=xe-xs:dy=ye-ys
100 laenge=SQR(dx*dx+dy*dy)
110 FOR entfernung =0 TO laenge STEP sw
120 PLOT xs+entfernung*dx/laenge,ys+entf
ernung*dy/laenge
130 NEXT punkt
```

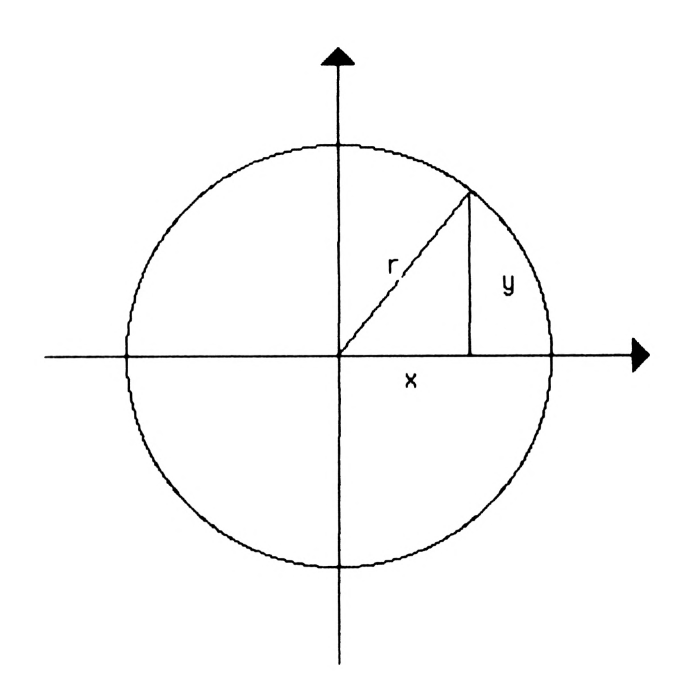
Auf diese Art und Weise lassen sich problemlos beliebig strichlierte Geraden zeichnen.

Kreise

Neben einfachen Linienzügen werden wir wohl am häufigsten Kreisbögen zur Gestaltung der verschiedensten Zeichnungen einsetzen.

Da das Locomotive Basic unseres CPC-464 leider keine einfach zu handhabende Circle-Anweisung für uns bereithält, kommen wir nicht umhin, uns auch über die mathematische Definition eines Kreises Gedanken zu machen.

So gilt als Kreis der geometrische Ort aller Punkte einer Ebene, die von einem festen Punkt dieser Ebene, dem Kreismittelpunkt, einen konstanten Abstand haben.
Weiterhin gilt, daß jede Strecke von diesem Mittelpunkt zu einem Punkt auf der äußersten Begrenzungslinie des Kreises, zur Kreisperipherie, als Radius bezeichnet wird.



Algebraisch läßt der Radius sich nach dem Satz des Pythagoras mit

r * r = x * x + y * y

r = SQR(x * x + y * y)

angeben, wenn der Mittelpunkt die Koordinaten (0,0) aufweist, die x- und die y-Achse sich also dort schneiden. Diese Gleichung kann nach y aufgelöst werden:

r * r = x * x + y * y

y = SQR(r * r - x * x)

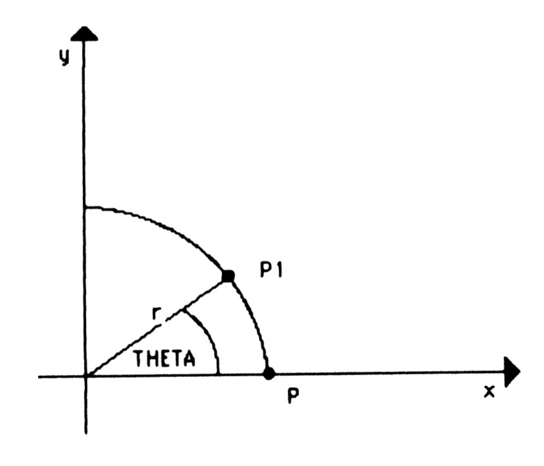
Um dann einen Kreis zeichnen zu können, müssen wir für x nacheinander alle Werte von -r bis r einsetzen, was sich programmtechnisch bestens durch eine Schleife realisieren läßt:

```
10 MODE 2
20 DEFINT A-Z
30 ORIGIN 320,200
40 INPUT "RADIUS ";RADIUS
50 FOR X=-RADIUS TO RADIUS
60 Y=SQR(RADIUS*RADIUS-X*X)
70 PLOT X,Y:PLOT X,-Y
80 NEXT X
```

In der Praxis bringt diese Routine jedoch einige Probleme mit sich, denn bei größeren Kreisradien sind die gewählten

Schrittweiten zu groß, um noch eine geschlossene Kreislinie zeichnen zu können.
Abhilfe schafft ein STEP .1 in Zeile 40, doch wird das Programm dann wegen des zehnfachen Rechnungsaufwandes entsprechend langsamer.

Besser eignet sich da schon ein anderes Verfahren, wenn wir nämlich nicht mehr die Lage eines jeden Punktes in Bezug auf die x-Achse berechnen, sondern uns stattdessen 'gradweise' auf der Kreislinie voranbewegen und direkt die x- und y-Koordinaten berechnen.
Dazu ist jedoch ein kleiner Vorgriff auf spätere Kapitel notwendig, denn wir müssen etwas Abstand von unserem gewohnten Koordinatensystem nehmen, und uns stattdessen den Polarkoordinaten zuwenden.
Bei dieser Art der Beschreibung der Lage eines Punktes in einer Ebene wird zum einen der Abstand des Punktes vom Koordinatenursprung, dem Pol, und zum anderen der Winkel zwischen der Abstandsgeraden und der positiven x-Achse angegeben.



Dieses Verfahren ist für unsere Zwecke natürlich ideal. So würden die Angaben für den Punkt p 0 Grad und 50 Längeneinheiten lauten und um die übrigen Punkte auf der Kreislinie angeben zu können, müßten wir nur die Gradzahlen um jeweils eins erhöhen bis wir bei 360 Grad angelangt sind.

Und da wir uns x/y -Koordinaten und nicht ominöse Winkelmaße wünschen, wenden wir dabei dann auch gleich zwei sogenannte Transformationsgleichungen an

$$x = r * \cos(\theta)$$

$$y = r * \sin(\theta)$$

und können auf diese Art und Weise jeden Kreis problemlos zeichnen:

```
10 MODE 2
20 DEFINT A-Z
30 ORIGIN 320,200
40 DEG
50 INPUT "RADIUS ";RADIUS
60 FOR GRAD=1 TO 360
70 X=RADIUS*COS(GRAD)
80 Y=RADIUS*SIN(GRAD)
90 PLOT X,Y
100 NEXT GRAD
```

Ellipsen

sind den Kreisen ja recht ähnlich, und so lassen sich die vorgestellten Routinen auch zur Darstellung von Ovalen und Ähnlichem einsetzen.

Um unsere Kreise zu strecken und zu stauchen, reicht es aus, unsere Transformationsgleichungen um einen Faktor zu ergänzen, wie folgende Demonstration zeigt:

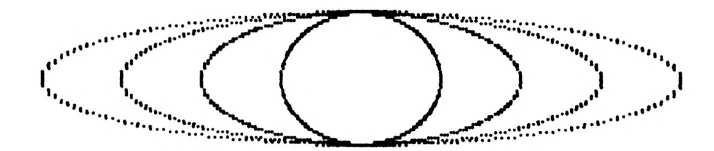
```
10 MODE 2
20 DEFINT A-Z
30 ORIGIN 320,200
40 DEG
50 INPUT "RADIUS ";RADIUS
60 FOR GRAD=1 TO 360
70 X=RADIUS*COS(GRAD)*VX
80 Y=RADIUS*SIN(GRAD)*VY
90 NEXT GRAD
```

VX und VY bewirken hierbei eine Verzerrung des Kreises in der entsprechenden Richtung.

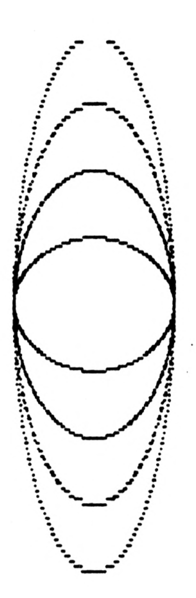
Allerdings sollten Sie bei Ihren Versuchen keine allzu großen Zahlen einsetzen; Werte zwischen Null und Zehn wären angebracht.

```
10 REM ellipsen
20 REM ________
30 MODE 2
40 DEFINT a-z
50 ORIGIN 320,200
60 DEG
70 IF w THEN CLS
80 CLS
90 INPUT "Radius ";radius
100 INPUT"Verzerrung in x- oder y-Richtu
ng (x/y)";richtung$
110 IF LOWER$(richtung$)="x"THEN vx=-1
120 IF LOWER$(richtung$)="y"THEN vy=-1
130 FOR grad=1 TO 360
140 FOR faktor=0.5 TO 3.5 STEP 0.5
 :REM verzerrungsgrad
150 x=radius*COS(grad)
160 IF vx THEN x=x*faktor
170 y=radius*SIN(grad)
180 IF vy THEN y=y*faktor
190 PLOT x,y
200 NEXT faktor
210 NEXT grad
220 FOR i=1 TO 10000:NEXT
230 w=-1:GOTO 70
```

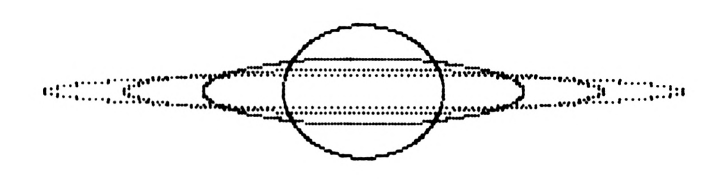
Radius: 50
x VX



Radius: 50
x VY



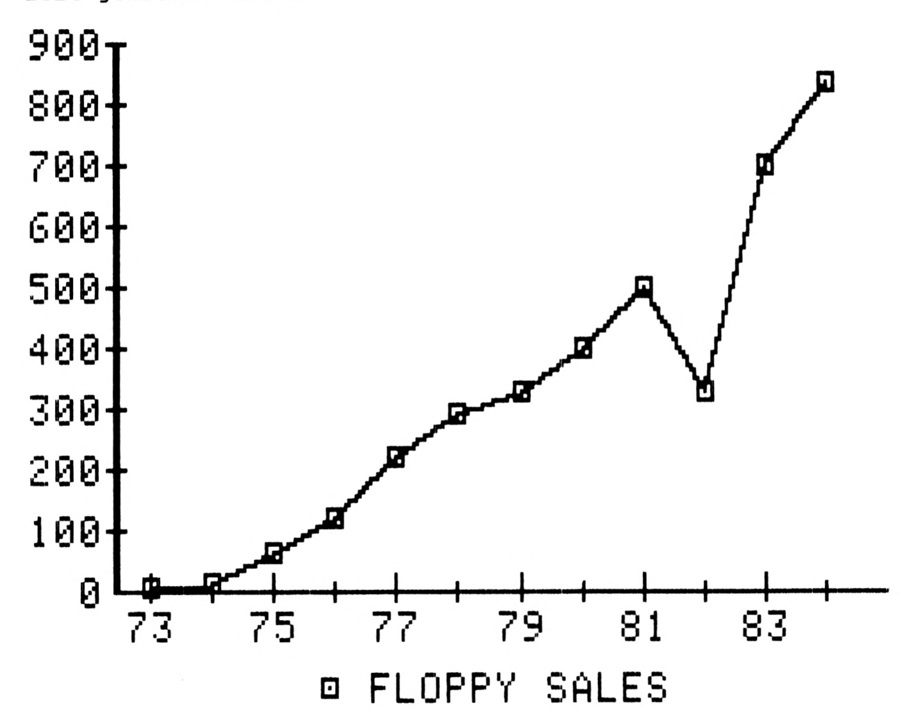
Radius: 50
x VX
/ VY



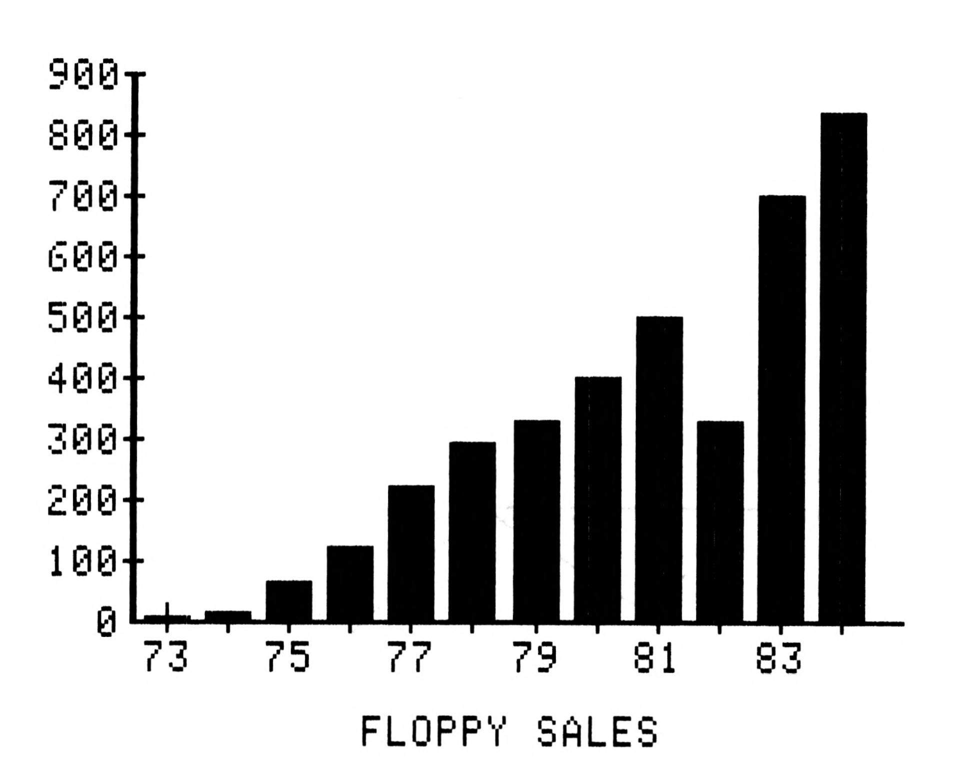
Anwendung: Kreisdiagramme

Da der Spruch 'Ein Bild sagt mehr als tausend Worte' auch im Geschäftsleben gilt, sind Firmenchefs immer schnell mit eindrucksvollen Grafiken zur Hand, wenn es darum geht, dem Geschäftspartner den eigenen Marktanteil zu verdeutlichen.
Je nach Zusammenhang und Aussageintention der Zahlenstatistik wird eine Darstellungsart gewählt welche die Verhältnisse ohne längeres Betrachten, Umdenken oder gar Rechnen deutlich macht.

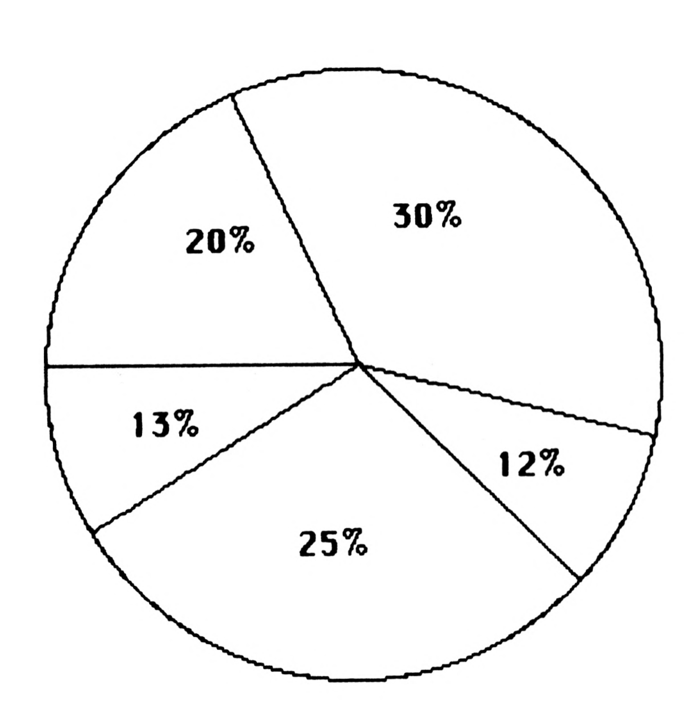
Bewährt haben sich sogenannte Kurvendiagramme, wenn Tendenzen sichtbar gemacht werden sollen.
Als Beispiel mag die Umsatzstatistik einer Firma gelten, bei der die Gesamtumsätze eines jeden Monats für den Zeitraum eines oder mehrere Jahre gegenübergestellt werden.
Aus dem Verlauf der Kurve können dann Rückschlüsse auf das Saisongeschäft gewonnen wie auch Vorhersagen für die nächste Zeit gemacht werden.



Balken- und Kreisdiagramme dienen weniger der Interpolation und Trendberechnung, als vielmehr der Gegenüberstellung von einem oder mehreren Datensätzen.
Dabei ist die Anzahl der einzelnen Meßwerte natürlich eingeschränkt - eine zehn Meter lange Wand mit fünfhundert Balken wäre zwar eindrucksvoll, aber wenig aussagefähig - weshalb diese Darstellungsart meistens für die Darstellung über größere Zeitspannen gewählt wird.
Aber auch hier ist der darzustellende Wert (Umsätze) wieder von einem anderen Faktor (Zeitraum) abhängig.



Ganz anders sieht es bei den Kreis- und Kuchendiagrammen aus.
Denn hier ist der größte Wert nicht durch die Zahlenstatistik gegeben, sondern es kann nur eine prozentuale Aufteilung der Kreisfläche stattfinden. Kreisdiagramme eignen sich daher besonders zur Gegenüberstellung von voneinander unabhängigen Werten.
Nicht umsonst heißt es, daß 'sich jeder sein Stückchen vom Kuchen abschneiden möchte', und so könnten beispielsweise die Umsatzzahlen von Konkurrenzfirmen Grundlage für diese Darstellung sein.



Doch welche Schritte werden nötig, um solch ein Kreisdiagramm auf dem Bildschirm entstehen lassen zu können?

Zum einen muß der Kreis erzeugt werden, dann müssen im richtigen Verhältnis zueinander Sektoren abgegrenzt werden und zu guter Letzt sollte auch noch eine Beschriftung der einzelnen Kreissegmente erfolgen, weil die Aussagekraft des Bildes sonst wohl doch recht gering sein dürfte.

Da die Daten auch noch irgendwie an das Programm übermittelt werden müssen, dürfte der grobe Aufbau wohl folgendermaßen aussehen:

Initialisierung

Dateneingabe

Plotten

Beschriften

Daß bei der Initialisierung der Speicherplatz für die Arbeitsvariablen bereitgestellt wird, und daß sich eine Reihe von gleichgestalteten Datensätzen am besten innerhalb einer Schleife eingeben lassen, dürfte jedem klar sein, der das Handbuch des Schneider CPC studiert hat, so daß sich zu folgendem Programmteil jeder Kommentar erübrigt:

```
30 MODE 1:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
40 DEG
50 DIM wert(20),grad(20)

1000 MODE 1:PRINT"Eingabe von Daten"
```

```
1010 PRINT:INPUT"Wie viele Einzeldaten "
;anz
1020 PRINT:ges=0
1030 FOR i=1 TO anz
1040 PRINT i;:INPUT wert(i)
1050 ges=ges+wert(i)
1060 NEXT i
```

Wie ein Kreis gezeichnet wird, weiß jeder, der die vorangehenden Seiten gelesen hat, und jeder, der schon einmal etwas von Dreisatzrechnung oder Verhältnisgleichungen gehört hat, weiß auch wie wir die einzelnen Kreisabschnitte berechnen.

Denn die 360 Grad eines Vollkreises entsprechen 100 Prozent und somit der Gesamtsumme aller eingegebenen Einzeldaten.
Ein einzelner Eingabewert wird somit soviele Grade von der gesamten Kreislinie in Anspruch nehmen, wie er an Einheiten zur Gesamtsumme beiträgt:

```
      Teilsumme           Teilgrad
 -------------------  =  --------
     Gesamtsumme          360 Grad
```

Somit muß der gesuchte Teilabschnitt (Teilsumme * 360 / Gesamtsumme) Grad betragen:

```
4000 MODE 2:ORIGIN 320,200
4030 FOR i=1 TO anz
4040 grad(i)=INT(wert(i)*360/ges)
4050 NEXT i
```

In dem Feld grad(i) sind nun die Kreislängen in Grad eines jeden Teilstückes gespeichert.

Alles, was wir nun noch tun müssen, ist, während des Plottens bei jedem Schritt zu prüfen, ob das Ende eines Sektors erreicht wurde (4100).
Wenn ja, dann wird zum einen eine Gerade zum Kreismittelpunkt gezeichnet, und zum anderen wird die bis zu diesem Zeitpunkt erreichte Gradzahl zum errechneten Endwert des nächsten Teilstückes addiert, so daß gewährleistet ist, daß auch wirklich nur das neue Teilstück berücksichtigt wird:

```
4060 r=150:i=1
4070 FOR grad=1 TO 360
4080 x=r*COS(grad):y=r*SIN(grad):PLOT x,
y
4100 IF grad(i)=grad THEN DRAW 0,0:grad(
i+1)=grad(i+1)+grad(i):i=i+1
4110 NEXT grad
4130 IF INKEY$="" THEN 4130
4140 GOTO 100
```

Die Beschriftung des Bildes könnte nun immer dann vorgenommen werden, wenn das Zeichnen eines Kreissektors abgeschlossen ist (4100), doch ist zum einen dann die Zuordnung recht zweideutig, und zum anderen sieht es auch nicht gut aus, wenn die Schrift mal ins Bild geht und mal daneben 'steht'.

Wünschenswert wäre die Benennung eines Sektors auf dessen halber Höhe; wir müssen somit bei der Berechnung der Gradanteile eines Sektors auch jeweils die Hälfte dieses Betrages für die Beschriftung gutschreiben:

```
4040 grad(i)=INT(wert(i)*360/ges):textpo
s(i)=grad(i)/2
4060 r=150:i=1:textalt=0
4090 IF textalt+textpos(i)=grad THEN GOS
UB 5000
```

Jeweils zum rechten Zeitpunkt kann dann ein Unterprogramm ab Zeile 5000 aufgerufen werden, welches zunächst die Länge des aktuellen Titels berechnet.
Anschließend wird geprüft, ob die Beschriftung in der linken Hälfte des Kreises erfolgen muß (x<0) oder ob das Label rechtsseitig ausgegeben wird.
Im ersten Fall wird der Grafikcursor um ein Leerzeichen und die Länge der Beschriftung nach links bewegt, im zweiten Fall nimmt der Cursor einen Abstand von eben diesem Leerzeichen von der Kreislinie ein:

```
5000 TAG:x1=(LEN(titel$(i))+1)*8
5010 IF x>0 THEN MOVER 10,0
5020 IF x<0 THEN MOVER -x1,0
5030 IF y>0 THEN MOVER 0,10
5040 PRINT titel$(i);
5050 TAGOFF
5060 RETURN
```

Da bei der Textausgabe mittels Grafikcursor immer mit der oberen linken Ecke eines Zeichens begonnen wird, erweist sich im Falle, daß y>0 ist, eine zusätzliche Verschiebung.

Beachten Sie bitte auch das Semikolon in Zeile 5050, dieses ist bei dieser Art der Textausgabe immer vonnöten, um Steuerzeichen wie Carriage Return, die sonst als Grafikzeichen erscheinen würden, zu unterdrücken.

Allerdings, bevor die Beschriftungen ausgedruckt werden können, müssen Sie diese dem Programm auch mitteilen. Wünschenswert wäre darüber hinaus in einigen Fällen wohl auch eine Möglichkeit, eingegebene Datensätze zu speichern. Ein dermaßen vervollständigtes Programm dürfte dann ähnlich dem folgenden Listing aussehen:

```
10 REM pieplot
20 DEFINT a-z
30 MODE 1:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
40 DEG
50 DIM wert(20),grad(20)
100 MODE 1:PRINT"      * * * *  PIEPLOT
* * * *":PRINT:PRINT" 1 - neuen Datensat
z eingeben"
110 PRINT" 2 - Datensatz speichern
120 PRINT" 3 - Datensatz laden
130 PRINT" 4 - Bild plotten"
140 PRINT:PRINT" Bitte waehlen Sie"
150 a$=INKEY$:IF a$="" THEN 150 ELSE a=V
AL(a$)
160 IF (a>4 OR a<1) THEN 100
170 ON a GOTO 1000,2000,3000,4000
990 REM ************ daten eingeben
1000 MODE 1:PRINT"Eingabe von Daten"
1010 PRINT:INPUT"Wie viele Einzeldaten "
;anz
1020 PRINT:ges=0
1030 FOR i=1 TO anz
1040 PRINT i,:INPUT wert(i):INPUT titel$
(i)
1050 ges=ges+wert(i)
1060 NEXT i
1070 PRINT:INPUT"Soll das Bild eine Uebe
rschrift tragen ";e$
1080 IF LEFT$(LOWER$(e$),1)="j" THEN INP
UT "Titel ";titel$
1090 GOTO 100
1990 REM *************** daten speichern
```

```
2000 CLS:PRINT"Ist ";LEFT$(titel$,8);" a
ls Dateiname okay ";:INPUT e$
2010 IF LEFT$(LOWER$(e$),1)="n" THEN INP
UT "Dateiname ";dn$
2020 OPENOUT dn$
2030 PRINT#9,titel$
2040 PRINT#9,anz
2050 FOR i=1 TO anz
2060 PRINT#9,wert(i):PRINT#9,titel$(i)
2070 NEXT i
2080 CLOSEOUT
2090 GOTO 100
2990 REM **************** daten laden
3000 CLS:INPUT "Dateiname ";dn$
3010 OPENIN dn$:ges=0
3020 INPUT#9,titel$
3030 INPUT#9,anz
3040 FOR i=1 TO anz
3050 INPUT#9,wert(i):ges=ges+wert(i):INP
UT#9,titel$(i)
3060 NEXT i
3070 CLOSEIN
3080 GOTO 100
3990 REM ******************* plotten
4000 MODE 2:ORIGIN 320,200
4010 IF anz=0 THEN PRINT" **** FEHLER !
- keine Daten.":FOR i=1 TO 10000:NEXT:GO
TO 100
4020 LOCATE 1,1:PRINT"Kurvendiagramm: ";
titel$
4030 FOR i=1 TO anz
4040 grad(i)=INT(wert(i)*360/ges):textpo
s(i)=grad(i)/2
4050 NEXT i
4060 r=150:i=1:textalt=0
4070 FOR grad=1 TO 360
4080 x=r*COS(grad):y=r*SIN(grad):PLOT x,
y
4090 IF textalt+textpos(i)=grad THEN GOS
UB 5000
4100 IF grad(i)=grad THEN textalt=grad:P
```

```
LOT x,y:DRAW 0,0:grad(i+1)=grad(i+1)+gra
d(i):i=i+1
4110 NEXT grad
4120 GOSUB 5000
4130 IF INKEY$="" THEN 4130
4140 GOTO 100
4990 REM **************** label
5000 TAG :x1=(LEN(titel$(i))+1)*8
5010 IF x>0 THEN MOVER 10,0
5020 IF x<0 THEN MOVER -x1,0
5030 IF y>0 THEN MOVER 0,10
5040 PRINT titel$(i);
5050 TAGOFF
5060 RETURN
```

Ausgefüllte Figuren

lassen sich mit den vorgestellten Routinen ebenfalls recht einfach zeichnen.
Haben wir bislang den Abstand einzelner Punkte vom Mittelpunkt des Kreises berechnet und diese Punkte dann gesetzt, um das Bild der Kreislinie zu erzeugen, so können wir mittels der Anweisung DRAW diese Entfernungslinie zeichnen und erhalten somit eine ausgefüllte Kreisfläche, in Computerkreisen häufig als 'DISC' bezeichnet.

Wenden wir probeweise das zuletzt entwickelte Verfahren an, so zeigt sich ein Mangel an Genauigkeit; das Verfahren nach Pythagoras erweist sich jedoch als hervorragend geeignet.
Denn hier wird ja die Kreislinie in kleinen Schritten entlang der x-Achse berechnet, womit das Resultat bei Anwendung von DRAW eine solide Fläche werden muß.

```
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
40 DEFINT A-Z
49 REM KOORDINATENURSPRUNG FESTLEGEN
50 ORIGIN 320,200
60 DEG

999 REM EINGABE KREISDATEN
1000 LOCATE 1,1:INPUT" RADIUS ";RADIUS
1310 FOR X=-RADIUS TO RADIUS
1320 Y=SQR(RADIUS*RADIUS-X*X)
1330 PLOT X,Y:DRAW X,-Y
1340 NEXT X
1360 GOTO 1000
```

Relativ genau und auch überraschend schnell erscheint die ausgefüllte Fläche eines beliebig großen Kreises auf dem Schirm.

Allerdings setzt der angewandte Algorithmus voraus, daß der Ursprung des Koordinatensystemes mit dem Mittelpunkt des Kreises identisch ist.
In der Computerpraxis ist es eigentlich üblich, mittels Addition jede Funktion innerhalb des Koordinatensystemes zu verschieben. Ein Programmierer des Schneider CPC sollte sich die Sache jedoch einfacher machen und für den Augenblick des Kreiszeichnens das Koordinatensystem nach dem Mittelpunkt des Kreises ausrichten.

Er darf nur nicht vergessen, bei Beendigung der Routine wieder die Standardeinstellung herbeizuführen (die wohl fast immer 0,0 oder 320,200 lauten wird):

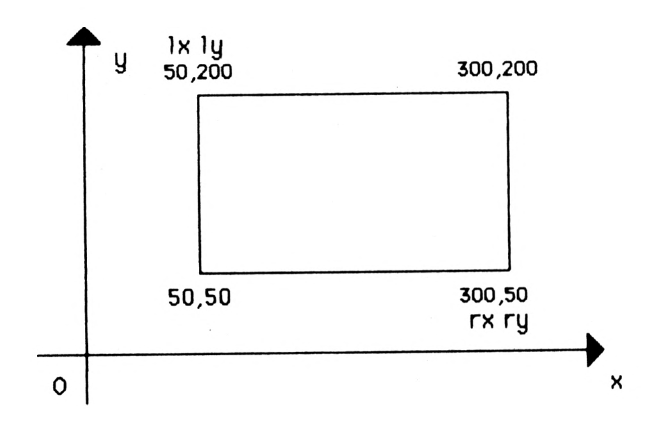
```
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
40 DEFINT A-Z
49 REM KOORDINATENURSPRUNG FESTLEGEN
50 ORIGIN 0,0
60 DEG

999 REM EINGABE KREISDATEN
1000 LOCATE 1,1:INPUT" RADIUS ";RADIUS
1010 INPUT" MITTELPUNKT (X,Y) ";MX,MY
1299 REM ------------------------- DISC
1300 ORIGIN MX,MY:REM NEUER URSPRUNG
1310 FOR X=-RADIUS TO RADIUS STEP 1
1320 Y=SQR(RADIUS*RADIUS-X*X)
1330 PLOT X,Y:DRAW X,-Y
1340 NEXT X
1350  ORIGIN  0,0      :REM  KOORDINATEN
WIEDERHERSTELLEN
1360 GOTO 1000
```

Auch Schraffuren werfen keine Probleme auf: die Breite des gewünschten Leerraumes kann mit STEP in Zeile 1310 festgelegt werden.
Nur eine Schrittweite von eins wird in sämtlichen Grafikbetriebsarten eine geschlossene Fläche erzeugen.

Rechtecke

Was den Kreisen recht ist, sollte den Rechtecken billig sein, doch fehlt uns bislang noch jegliche Methode, um diese möglichst einfach zu erzeugen.
Natürlich, vier Linien im rechten Winkel zueinander angebracht, führen auch zum Ziel, doch warum sollte sich jemand mit vielen Koordinaten abmühen, wenn es auch anders geht ?



Wie man sieht, lassen sich die Koordinaten der fehlenden Punkte mit (lx,ry) und (rx,ly) angeben.
Und weil die Sache so einfach ist, haben viele Hersteller den Basic-Dialekt ihres Computers um einen BOX-Befehl ergänzt.
Dieser bringt nach Angabe der linken oberen und der rechten unteren Ecke in Windeseile das gewünschte Rechteck auf den Bildschirm.

Unser Scheider kennt solch einen Befehl nicht, doch macht uns ein kleines Unterprogramm diese Vorgehensweise ebenfalls möglich:

```
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
40 DEFINT A-Z
49 REM KOORDINATENURSPRUNG FESTLEGEN
50 ORIGIN 0,0
1399 REM RECHTECK ZEICHNEN
1400 LOCATE 1,1:INPUT " LINKS OBEN  (X,Y
)";LX,LY
1410 INPUT" RECHTS UNTEN (X,Y)";RX,RY
1440 PLOT LX,LY:DRAW RX,LY:DRAW RX,RY:DR
AW LX,RY:DRAW LX,LY
1490 GOTO 1400
```

Da der Videoteil des Schneider CPC speziell für Grafik ausgelegt wurde, ist die Geschwindigkeit dabei noch so groß, daß optisch kein Unterschied zu einem Computer mit einer entsprechenden Interpreterroutine ausgemacht werden kann.
Wie Sie sich selbst überzeugen können, spielt es auch keine Rolle, welchen Punkt Sie zuerst eingeben.
Sie müssen nur beachten, daß die Eckpunkte sich diagonal gegenüberstehen.

Um eine FILLED BOX, ein ausgezeichnetes Rechteck, auf den Bildschirm zu bekommen, können wir uns zweier Verfahren bedienen.

Zum einen gehen wir wie bei den Kreisen vor, indem wir an der einen Achse entlangschreiten und Linien über die festgelegte Distanz in Richtung der anderen Achse zeichnen:

```
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
40 DEFINT A-Z
49 REM KOORDINATENURSPRUNG FESTLEGEN
50 ORIGIN 0,0
1399 REM RECHTECK ZEICHNEN
1400 LOCATE 1,1:INPUT " LINKS OBEN  (X,Y
)";LX,LY
1410 INPUT" RECHTS UNTEN (X,Y)";RX,RY
1420 INPUT" AUSFUELLEN   (J/N)";E$
1430 IF UPPER$(E$)="J" THEN 1500
1440 PLOT LX,LY:DRAW RX,LY:DRAW RX,RY:DR
AW LX,RY:DRAW LX,LY
1490 GOTO 1400
1499 REM FILLED BOX
1500 REM
1510 FOR Z=LX TO RX STEP 1
1520 PLOT Z,LY:DRAW Z,RY
1530 NEXT Z
1550 GOTO 1400
```

Solange wir uns an die Reihenfolge links oben/rechts unten halten, funktioniert alles recht gut, nur umgekehrt tut sich leider gar nichts.

Den Grund für dieses Verhalten finden wir in der Konstruktion der Schleife, denn diese müßte die x-Werte nun laufend verringern anstatt sie zu vergrößern.

Als Ausweg bietet es sich an zu prüfen, ob der zweite Wert denn wirklich größer als der erste ist, und diese beiden Zahlen dann gegebenfalls zu vertauschen.

Ergänzen Sie obige Zeilen daher bitte um:

```
1500 IF LX>RX THEN HX=RX:RX=LX:LX=HX
```

Der zweite Weg, von dem eben die Rede war, macht sich die besondere Technik der Windows zunutze, die der CPC bekanntermaßen meisterhaft beherrscht.

Windows

entstammen einer Zeit, als gewiefte Programm- und Computerhersteller darüber nachdachten, wie sie einer technisch völlig unbedarften Sekretärin den Umgang mit dem neuen Kollegen 'Computer' vereinfachen konnten.
Denn meistens war es so, daß die Mitarbeiter der Firma auch nach der Einarbeitung mehr Zeit mit dem Wälzen der Handbücher statt der Büroarbeit verbrachten.
So entstand der Gedanke, notwendige Anweisungen einfach auf dem Bildschirm einzublenden, sie gleich einem Blatt auf das Arbeitspapier zu legen.

Und so, wie man ein Blatt wieder beiseitelegen kann, so sollten sich diese Hinweise auch wieder aus dem Bild ausblenden lassen.

Um in dieser vollendeten Form in Erscheinung treten zu können, ist bereits ein stattlicher Programmieraufwand erforderlich, ebenso steigt der Speicherbedarf des Computersystemes rapide an: schließlich muß der alte Bildschirminhalt immer erst abgespeichert werden, bevor ein neues Fenster über ein altes gelegt werden kann.
Ist der Bildschirmspeicher dann ähnlich dem des CPC aufgebaut, dann müßten bei nur vier übereinandergelegten und später einzeln restaurierbaren Fenstern bereits 48k an Speicherplatz zur Verfügung stehen.
Aus diesem Grunde finden wir derartige Anwendungen der Window-Technologie nur bei Systemen, die mit einem schnellen Massenspeicher großer Kapazität, sprich einer Festplatte, oder mit einem Speicher von mehreren 100 kByte ausgestattet sind.

Beides trifft nun leider auf unseren CPC in der üblichen Konfiguration nicht zu, dennoch lassen sich mit den WINDOWs des Locomotive-Basic auch schon interessante Anwendungen programmieren.

Obwohl es sich bei den Fenstern auch immer um Rechtecke handelt, unterscheiden sie sich in ihrer Definition doch ein wenig von unserem obigen Verfahren.
Denn nach dem Schlüsselwort WINDOW und nach # ist mit einer Zahl zwischen null und sieben dem Fenster erst eine sogenannte Kanalnummer zuzuweisen, über das es dann später immer wieder angesprochen werden kann.
Anschließend werden die Eckpunkte des Fensters in der Reihenfolge links, rechts, oben, unten angegeben - und zwar nach Schreibstellen auf dem Bildschirm.
So kommt es, daß ein Fenster mal von eins bis 22 (Mode 0) oder auch mal bis 80 (Mode 2) gehen kann.

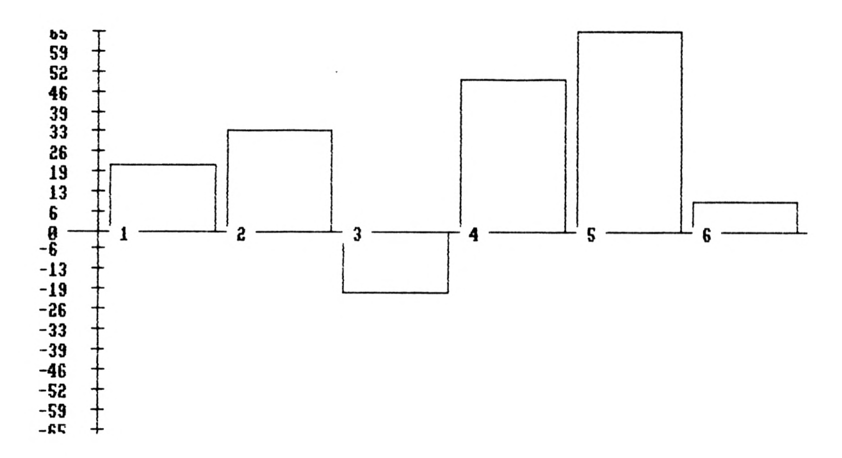
Eine dem Printbefehl mittels # nachgestellte Nummer

adressiert nun ein zuvor unter dieser Ziffer festgelegtes Fenster, aber auch an den Clear Screen Befehl kann die Kanalnummer angehängt werden:

```
10 MODE 0
20 RECHTS=22:UNTEN=24
30 FARBE=2
40 FOR KANAL=0 TO 7
50 LINKS=LINKS+2:OBEN=OBEN+2
60 WINDOW #KANAL,LINKS,RECHTS,OBEN,UNTEN
70 FARBE=FARBE+1
80 PAPER #KANAL,FARBE
90 CLS #KANAL
100 FOR I=1 TO 400:NEXT
110 NEXT
```

Anwendung: Balkendiagramm

Es ist nun an der Zeit, unser Programm Pieplot um die Möglichkeit zur Erstellung von Balkendiagrammen, sogenannten Bar-Charts, zu erweitern.



Prinzipiell stellen sich uns die gleichen Probleme wie zuvor, denn die Daten müssen wiederum erfaßt und vor dem eigentlichen Zeichenvorgang an die Zeichenfläche angepaßt, sie müssen skaliert werden.

Diese Skalierung ist einer der wichtigsten Arbeitsvorgänge, der bei Darstellungen gleich welcher Art berücksichtigt werden muß, weshalb wir diesen Vorgang an dieser Stelle genauer untersuchen wollen.

Tatsächlich findet bei dieser Maßstabsfindung zunächst immer ein Vergleich statt, denn entweder werden die Datensätze alle nach einer oberen Grenze ausgerichtet, wie es auch schon bei den Kreisdiagrammen der Fall war (obere Grenze gleich 360 Grad), oder aber der größte darzustellende Meßwert wird als Bezugsgröße eingeführt und alle übrigen Daten werden nach diesem Maximum ausgerichtet.

Unter dem Aspekt einer möglichst großen Darstellung der Zahlenverhältnisse auf der Zeichenfläche wird man nun diesen Maximalwert der Daten dem Maximalwert der größtmöglichen Zeichenflächenausdehnung gleichsetzen:

$$\frac{\text{Plotmax}}{\text{Datamax}}$$

Der daraus resultierende Quotient ist ein Skalierungsfaktor, welcher durch Multiplikation mit jedem einzelnen Datenwert diesen an die Darstellung des größten Wertes, und somit an die Zeichenfläche, anpaßt.
Das Produkt aus Datenwert * Skalierungsfaktor kann somit bereits als direkte Entfernungsangabe des darzustellenden Wertes in Bezug auf die Nullinie der Zeichenfläche aufgefaßt werden.

Da die Koordinaten dieser Grundlinie zu Beginn des Zeichenvorganges gewählt, die darzustellende Entfernung von dieser Grundlinie berechnet wird und zur Darstellung eines Rechteckes nur zwei Punkte bekannt sein müssen, ist das Problem bereits gelöst, und das zuvor erstellte Programm PIEPLOT kann erweitert werden:

```
50 DIM wert(20),grad(20),bwert(20)

100 MODE 1:PRINT"      * * * *  DATAPLOT
 * * * *":PRINT:PRINT" 1 - neuen Datensa
tz eingeben"
135 PRINT" 5 - Balkendiagramm"
160 IF (a>5 OR a<1) THEN 100
170 ON a GOTO 1000,2000,3000,4000,6000
```

Diese Zeilen stellen den benötigten Variablenspeicherplatz bereit und erweitern das Menü um die Funktion Balkendiagramm.

In Zeile 1050 wird eine Ergänzung notwendig, welche den größten Eingabewert als wmax notiert, so daß im folgenden der Skalierungsfaktor wie erläutert berechnet werden kann:

```
1050 ges=ges+wert(i):wmax=MAX(wert(i-1),
wert(i))

6000 MODE 2:mfaktor=199/wmax:rx=60:absta
nd =10
```

Vielleicht wundert es Sie nun, warum bei der Berechnung des Skalierungs-, der hier als Maßstabsfaktor bezeichnet wird, nur eine maximale Ausdehnung der Zeichenfläche von 199 und nicht 400 (Punkte in y-Richtung) angegeben werden.

Sie sollten jedoch bedenken, daß zum einen in solchen Diagrammen auch die Möglichkeit zur Darstellung negativer Werte gewahrt bleiben sollte, weshalb der Maximalwert sich halbiert, und zum anderen könnte es geschehen, daß die obere Begrenzungslinie nicht mehr gezeichnet wird, weil sich durch die Multiplikation mit mf ein etwas größerer Wert als 200 ergibt.

Im Sinne einer späteren Beschriftung der y-Achse wäre es sogar sinnvoll, die obere Grenze noch etwas niedriger anzusetzen.

Unter Beachtung dieser Punkte kann die x-Achse somit nur in der Mitte des Schirmes liegen; die y-Achse wird ein klein wenig von der linken Zeichenflächengrenze abgesetzt, so daß noch genügend Platz für eine spätere Beschriftung bleibt:

```
6060 ORIGIN 0,200
6070 PLOT 10,0:DRAW 640,0:REM x-achse
6080 PLOT 50,-200:DRAW 50,200:REM y-ach
se
```

Bevor die einzelnen Balken dann gezeichnet werden, muß innerhalb einer Schleife deren Länge berechnet werden, außerdem wird ihre Breite in Abhängigkeit von ihrer Anzahl, der zur Verfügung stehenden Bildfläche und dem zwischen ihnen liegenden Abstand abhängig sein:

```
6030 FOR i=1 TO anz
6040 bwert(i)=wert(i)*mfaktor
6050 NEXT i

6090 balkenbreite=580/anz-abstand
```

Die Balken selbst werden dann mit einer der zuvor entwickelten Routinen dargestellt:

```
6100 FOR i=1 TO anz
6110 ry=0:lx=rx+balkenbreite:ly=bwert(i)
6120 PLOT lx,ly:DRAW rx,ly:DRAW rx,ry:DR
```

```
AW lx,ry:DRAW lx,ly
6140 rx=rx+abstand+balkenbreite:REM begi
nn des naechsten Balkens
6150 NEXT i
```

Auf jeden Fall sollte anschließend auch die y-Achse beschriftet werden, dies geht jedoch nicht ohne einige weitere Vorüberlegungen und Umrechnungen.

Denn da nicht nur einfach gleichmäßige Abstände markiert werden sollen, sondern diese entsprechend der dargestellten Daten beschriftet werden müssen, ergibt sich daraus schon eine Multiplikation mit dem Skalierungsfaktor.
Dabei ist der Wert des größten dargestellten Punktes mit wmax identisch, weshalb ein Teilstück so groß wie wmax dividiert durch die gewünschte Anzahl dieser Teilstücke, hier zehn, sein muß:

```
6010 TAG
6220 FOR y1=0 TO wmax STEP wmax/10
6230 PLOT 55,y1*mfaktor:DRAW 45,y1*mfakt
or
6240 MOVE 1,y1*mfaktor+4
6250 PRINT INT(y1);
6250 NEXT y1
```

Entsprechend kann auch der negative Teil der y-Achse beschriftet werden:

```
6170 FOR y1=-wmax+1 TO 0 STEP wmax/10
6180 PLOT 55,y1*mfaktor:DRAW 45,y1*mfakt
or
6190 MOVE 1,y1*mfaktor+4
```

```
6200 PRINT INT(y1);
6210 NEXT y1
6270 TAGOFF
6280 GOTO 4130
```

Wer nun auch noch eine Beschriftung der x-Achse wünscht, kann das Programm um folgende Zeile ergänzen:

```
6130 MOVE rx,ry+4:PRINT i;
```

Ihnen steht nun ein Programm zur Verfügung, welches eine Reihe von Daten als Kreis- oder Balkendiagramm darstellt.
Dabei werden Skalierungen automatisch in der Form vorgenommen, daß die Daten immer formatfüllend dargestellt werden.

Dennoch, die physikalisch festgesetzten Werte des Bildschirmes kann kein noch so gutes Programm umgehen, versuchen Sie daher nicht, allzu viele Einzeldaten grafisch darzustellen.

3. KAPITEL
GRAFIK-TECHNIKEN

Die Bildschirmtechnik

entscheidet oftmals über die Eignung eines Computers zu den verschiedensten Anwendungsmöglichkeiten im Bereich der Computergrafik.
So kann eine möglichst hohe Auflösung zur Darstellung mathematischer Zusammenhänge gewünscht sein, wobei das erzeugte Bild ruhig einfarbig dargestellt werden darf, oder aber es wird eine farbige Darstellung, die vielleicht nicht ganz so exakt, aber dafür schnell auf den Monitor zu bringen sein muß, verlangt, wie es bei den Action-Computerspielen der Fall ist.

Aus diesem Grunde ist man bestrebt, die vorhandene Hardware möglichst gut zu nutzen und mit einem Mindestmaß an zeitaufwendigen Berechnungen auszukommen.
So wird sich auch der CPC 464 nach einem PRINT "H" schneller mit READY zurückmelden, als wenn Sie drei einzelne Linien plotten lassen, welche das gleiche Bild ergeben.
Der zeitliche Unterschied kommt dadurch zustande, daß das Betriebssystem sich im ersten Fall nur das 'Bitmuster' des Zeichens aus einer bestimmten Stelle des Speichers holen muß, während bei der zweiten Methode drei Plot und drei Draw-Befehle ausgeführt werden müssen.

Drei verschiedene Techniken sind es nun, die sich im Laufe der letzten Jahre hervorgetan haben und sich alle die Benutzung vorgefertigter Zeichen zu eigen machen.

Sprites

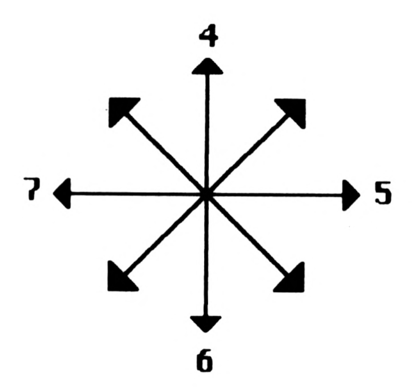
sind das jüngste Kind dieser Entwicklung und wohl in der Hauptsache für den riesigen Erfolg des Heimcomputers Commodore 64 verantwortlich zu machen.

In diesem Computer, der auf Telespiele in Spielhallenqualität ausgelegt ist, ist ein Grafikprozessor eingebaut, welcher es dem Anwender möglicht macht, eine Reihe von vorgefertigten Figuren zu erzeugen und zu bewegen, die bis zu 24 x 21 Punkte groß sein können.

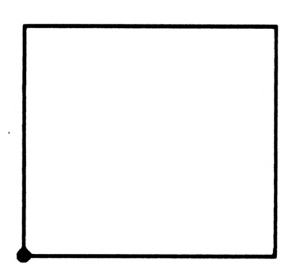
Der wesentlichste Vorteil der Sprites besteht darin, daß sie sich über den Bildhintergrund bewegen lassen, ohne diesen zu zerstören, das heißt ohne irgendetwas zu überschreiben oder zu löschen.

Shapes

sind die Vorfahren der Sprites. Ihre Erscheinungform ist jedoch immer stark vom Hersteller des jeweiligen Computers abhängig, im Prinzip handelt sich dabei aber immer um die Umrisse einer darzustellenden Figur.
Diese werden dann als Richtungsanweisungen für den Cursor innerhalb einer Tabelle, der Shape-Table, notiert.



Es gibt somit eine bestimmte Kennzahl für eine Bewegung nach links und nach rechts, nach oben und nach unten; häufig sind auch Codes für diagonale Bewegungen vorhanden wie auch noch unterschieden wird, ob während der Bewegung gezeichnet werden soll oder nicht.



Wenn außerdem ein Anfangspunkt vereinbart wurde, vielleicht in der Form, daß vereinbart wurde, daß das Zeichnen der Figur immer unten links beginnt, dann kann obiges Quadrat in der Form

4,5,6,7

beschrieben werden.

Zwar bietet diese Methode nicht unbedingt einen Geschwindigkeitsvorteil, doch läßt dieses Konzept Manipulationen wie Vergrößern und Rotieren zu.
Aus diesem Grunde werden wir im nächsten Kapitel auch eine Methode einführen, wie die Shapetechnik auch auf dem Schneider CPC angewandt werden kann.

Charakters

Wenn die Shapes die Eltern der Sprites sind, dann sind die Charakters und die Kombinationen derselben, die Strings, in dieser Hinsicht soviel wie Adam und Eva bei den Menschen, nämlich der Ursprung allen Seins und aller Technik.

Denn in der Computerurzeit war man ja schon froh, wenn der Computer überhaupt mit einem Drucker versehen war und nicht nur Lochkarten oder Lochstreifen ausgab, welche erst noch dekodiert werden mußten.
Und da diese Drucker eher mißbrauchte Fernschreiber denn das, was heute diesen Namen trägt, waren, war auch der Zeichenvorrat recht beschränkt.

Doch wurden natürlich auch damals schon Funktionsgraphen zu Papier gebracht.
Allerdings war der kleinste Punkt dieser Grafiken dann immer noch so groß wie ein Buchstabe - egal, ob es sich nun um ein Sternchen oder ein Interpunktionszeichen handelte.

Erst 1953 wurde der erste computergesteuerte Punkt auf einem Bildschirm erzeugt und die Grundlage zu der heute üblichen Buchstabendarstellung gelegt.
Denn die darzustellenden Bilder, und dementsprechend auch die Ziffern und Zeichen, wurden in mehrere kleine Punkte aufgeteilt, die der Elektronenstrahl nacheinander zum Leuchten bringen konnte.

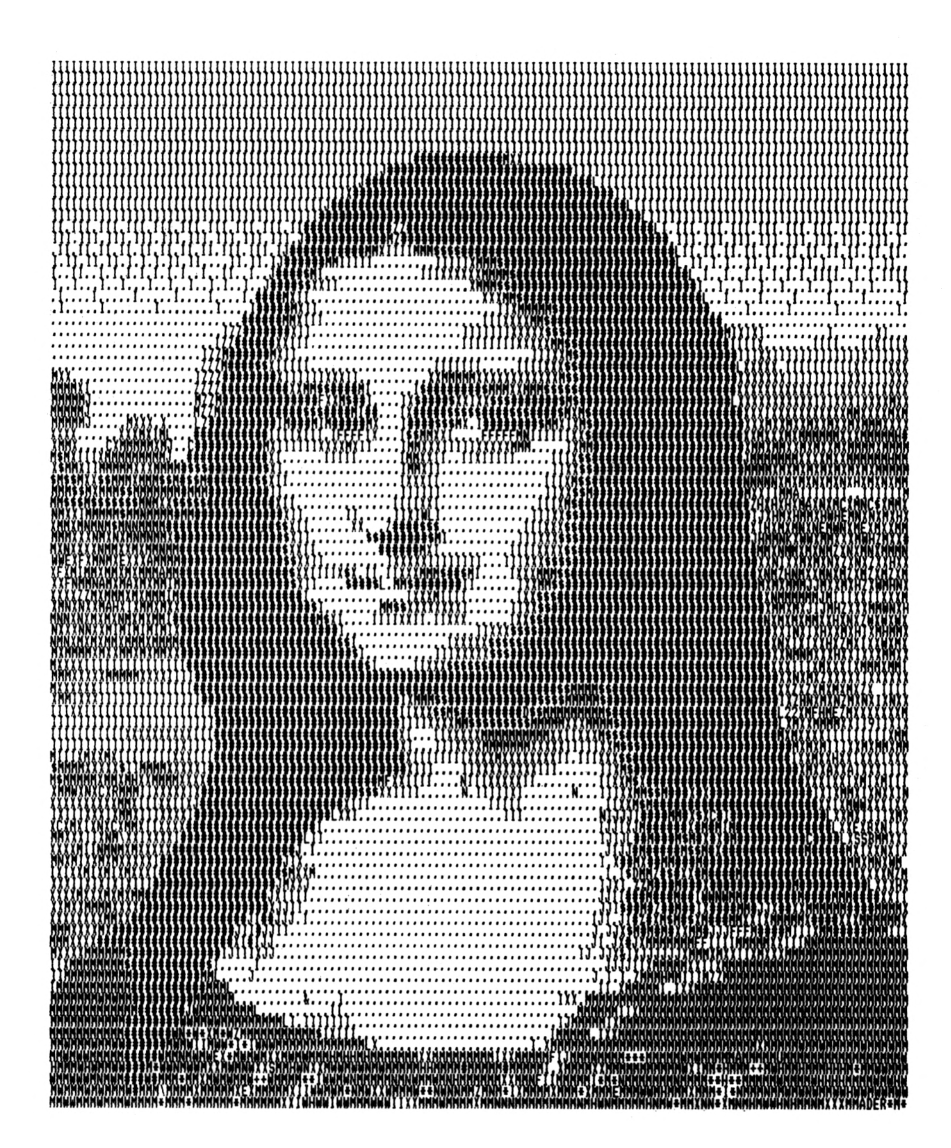
Diese sogenannte Rastergrafik konnte bis heute nicht durch ein anderes Verfahren ersetzt werden, Verbesserungen wurden nur durch die Benutzung mehrerer Punkte zur Darstellung eines Zeichens erzielt.

Üblich im Heim- und Hobbybereich ist dabei ein Raster von 8 x 8 Punkten, so daß jedes Zeichen aus maximal 64 Punkten bestehen kann.

Damit wurde ein recht guter Kompromiß geschlossen, denn die höheren Auflösungen erfordern, da die Bitmuster eines jeden Zeichens irgendwo gespeichert werden müssen, mehr Speicherplatz und zusätzlichen technischen Aufwand.
Und auch die verwendete Bildröhre muß in der Lage sein, diese Vielzahl von Punkten scharf abzubilden - wobei man den Aspekt der Schärfe allerdings auch nicht zu genau nehmen darf.
Denn sonst bleiben weiterhin die einzelnen Punkte sichtbar und es entsteht nicht der Eindruck eines geschlossenen Linienzuges.

Jahrelang war man also genötigt, die 'eingebauten' Buchstaben nicht nur zur Textbe- und verarbeitung zu verwenden, sondern auch zur Darstellung der unterschiedlichsten Grafiken.
Einige frühe Computerkünstler haben sich sogar die Mühe gemacht, Photografien in Buchstabenbilder umzusetzen, und diese per Hand zu digitalisieren.
Dabei machten sie sich den Umstand zunutze, daß ein 'm' wesentlich dichter ist, also aus mehr gesetzten Punkten besteht, als ein 'c' oder gar ein Punkt.

Durch geschickte Buchstabenkombinationen ließen sich so Helligkeitsabstufungen simulieren, welche die so konstruierten Bilder bei ausreichend großem Betrachtungsabstand recht passabel aussehen ließen.



Doch war eine so erstellte Grafik noch viel zu ungenau, wenn es um technische Dinge ging, weshalb die Hersteller den Zeichensatz ihrer Geräte um Sonderzeichen erweiterten.

Dies waren in der Regel Linien, Winkel, Kreisbögen und unterschiedlich große Rechtecke und Quadrate wie auch Gruppierungen derselben.

Solche Grafikzeichen läuteten das Zeitalter der animierten Computerspiele ein, und PONG, ein dem Tischtennis nachempfundenes Telespiel, dessen grafische Bestandteile, nämlich Schläger, Ball und Spielfeldbegrenzung, nur aus solchen rechteckigen Blöcken bestanden, war nach seiner Vorstellung im Jahre 1974 bald in jeder Gaststätte und Imbißstube zu finden.

Blockgrafiken

in diesem Stil sind auch heute noch der wichtigste Bestandteil aller Computerspiele, stehen diese Grafikzeichen doch in relativ großer Vielfalt zur Verfügung und lassen sich außerdem ohne besonderen Hard- und Softwareaufwand benutzen.

Doch war es schon ärgerlich, wenn man eine im Winkel von 22.5 Grad ansteigende Gerade darstellen wollte, der Zeichensatz des Computers aber nur über eine 45 Grad Diagonale verfügte.
Wunschlos glücklich wären die Programmierer jener Tage gewesen, wenn sie die Grafikzeichen selber hätten definieren können. Eine Möglichkeit, die heute im Zeitalter der einzeln ansprechbaren Punkte gerne als veraltet abgetan wird.

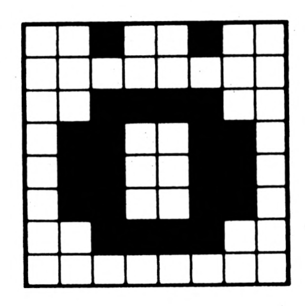
Wie glücklich sollte dann ein Schneider - Anwender sein,

kann er doch nicht nur Punkte einzeln setzen und Linien zeichnen, eingebaute Text- und Grafikzeichen nutzen, sondern auch sämtliche Zeichen redefinieren - eine eigene Codesprache wäre wirklich kein Problem.

Dabei sind nicht einmal Umrechnungsarbeiten nötig, auch das erledigt Ihr Computer, wie wir Ihnen am Beispiel des Umlautes 'ö' zeigen wollen.

Zeichendefinition

Nehmen Sie sich zunächst etwas kariertes Papier und markieren Sie eine 8 x 8 Kästchen große Fläche.
Anschließend könnten Sie versuchen, das ö oder jedes andere gewünschte Zeichen frei aus der Hand in dieses Quadrat zu zeichnen.
Um dann einen Eindruck zu gewinnen, wie die Creation sich auf dem Bildschirm macht, sollten Sie die Kästchen, aus denen Ihr Buchstabe im Endeffekt bestehen soll, ausmalen.



Anschließend sollten Sie neben jede Reihe eine Gruppe von 'Einsen' und 'Nullen' schreiben, wobei eine 1 einem ausgefüllten, und eine Null entsprechend einem leeren Kästchen zugeordnet wird.
Diese so entstandenen 'Binärzahlen' sind das Bitmuster, welches die Hardware konstruktionsmäßig auf dem Bildschirm zum leuchten bringt, und die Sie dem Betriebssystem in der Form von Dezimalzahlen mitteilen müssen.

Das hört sich nun doch schon wieder nach Rechenarbeit an und kann in einer Zeit, in der jeder Schüler automatisch seinen Taschenrechner benutzt, bereits ein gewichtiger Grund gegen selbstdefinierte Zeichen sein.

Und falls dann doch noch jemand seine zehn Finger benutzen sollte und sich nicht ins Zweiersystem einarbeiten möchte, dann kann er mit PRINT &X 00111100 diese Arbeit auch an seinen CPC delegieren.
Selbstverständlich kann dabei anstelle von 00111100 auch jede andere Binärzahl eingesetzt werden kann.

00100100
00000000
00111100
01100110
01100110
01100110
00111100
00000000

Stehen die acht Dezimalzahlen dann fest, ist die Entscheidung zu treffen, welches Zeichen durch diese Gruppierung ersetzt werden soll. Anschließend kann der Vorgang mit einem SYMBOL x,w1,w2,w3,w4,w5,w6,w7,w8 abgeschlossen werden, wobei x für die Codenummer (CHR$) des zu ändernden Zeichens steht und w1 bis w8 die errechneten Dezimalwerte sind.
Nicht vergessen werden darf allerdings zuvor die Anweisung SYMBOL AFTER x, womit x einen Charaktercode festlegt, der überschritten werden muß, wenn ein Zeichen geändert werden soll.

Da, wie wir im folgenden noch sehen werden, die Grafikzeichen im besonderen und die Buchstaben im allgemeinen so immens wichtig werden, wollen wir an dieser Stelle nun ein Programm vorstellen, welches Ihnen diese Arbeit des Berechnens und Umdefinierens abnimmt und ähnlich dem Programm auf der dem Schneider CPC beigefügten Demonstrationskassette arbeitet.

Anwendung: Zeichensatz-Editor

Im wesentlichen wird das Programm aus zwei Teilen bestehen:

1. Gestaltung des neuen Zeichens auf dem Bildschirm
2. Auswertung und Redefinition

Denn es wäre wenig sinnvoll, das Zeichen immer noch auf dem Papier 'per Hand' zu konstruieren und dem Computer nur die Rechenarbeit zu überlassen.
Wir wollen daher zunächst eine 8 x 8 Einheiten große Fläche auf dem Bildschirm abgrenzen, und den Cursor dann gezielt innerhalb dieses Feldes bewegen.

Sich merken, ob ein einzelner Punkt des Zeichens nun gesetzt oder gelöscht ist, kann der Computer am besten innerhalb eines zweidimensionalen Feldes:

```
10 DIM buchstabe$(8,8)
20 FOR reihe=1 TO 8
30 FOR spalte=1 TO 8
40 buchstabe$(reihe,spalte)=CHR$(144)
50 NEXT spalte,reihe
```

Da die obere linke Ecke des Bildschirmes als Printposition 1,1 definiert ist, bereitet auch die Ausgabe dieses Feldes auf dem Monitor keine Schwierigkeiten:

```
70 GOSUB 500
500 LOCATE 1,1:FOR re=1 TO 8
510 FOR sp=1 TO 8
520 PRINT buchstabe$(re,sp);
```

```
530 NEXT sp
540 PRINT
550 NEXT re
570 RETURN
```

Somit entspricht die Darstellung auf dem Bildschirm genau der Anordnung der einzelnen Daten innerhalb des Feldes, weshalb die Reihen- und Spaltenposition auch als Adressierung des Cursors benutzt werden kann.
Dabei soll die Bewegung der Schreibmarke durch Drücken einer Taste des 10'er Blocks in Richtung der Anordnung dieser Taste erfolgen, die Nummer der Reihe/Spalte wird sich also um eins erniedrigen/erhöhen:

```
80 reihe=1:spalte=1
100 eingabe$=INKEY$
110 IF eingabe$="8" THEN reihe=reihe-1
120 IF eingabe$="2" THEN reihe=reihe+1
130 IF eingabe$="6" THEN spalte=spalte+1
140 IF eingabe$="4" THEN spalte=spalte-1
150 IF eingabe$="7" THEN reihe=reihe-1:s
palte=spalte-1
160 IF eingabe$="9" THEN reihe=reihe-1:s
palte=spalte+1
170 IF eingabe$="3" THEN reihe=reihe+1:s
palte=spalte+1
180 IF eingabe$="1" THEN reihe=reihe+1:s
palte=spalte-1
```

Übrigens erfolgt so wie hier in diesem Beispiel des Zeicheneditors jegliche Cursorsteuerung, egal ob es sich nun um die Steuerung der Schreibmarke selbst oder aber um ein Raumschiff handelt.

Immer wird zunächst eine Anfangskoordinate (hier 1,1) festgelegt, welche dann entsprechend der vorgesehenen Tastensteuerung manipuliert wird.
Selbstverständlich können Sie sich dabei dann auch auf Grafikpositionen beziehen, und ein Shape, oder ein Zeichen mittel TAG, an jede gewünschte Stelle bewegen, wobei der Summand die Geschwindigkeit der Bewegung festlegt.

Doch gerade im Bereich der Spiele, vielleicht aber auch bei dieser Anwendung, werden Sie es für wünschenswert halten, den Cursor nicht mittels der Tastatur, sondern über einen angeschlossenen Joystick zu bewegen.

Erinnern Sie sich noch an unsere Ausführungen zu den Shapes?

Ähnlich verhält sich auch der Joystick, denn er übermittelt dem Rechner eine Zahl, die einer genau definierten Richtung oder Funktion entspricht:

1 - nach oben
2 - nach unten

4 - nach links
8 - nach rechts

16 - Taste II
32 - Taste I

Die Werte für die zusammengesetzten Richtungen, die Diagonalen, ergeben sich durch Addition der Werte für die Teilrichtungen.
Wird außerdem bei der Bewegung des Joysticks in eine

Richtung der 'Feuerknopf' niedergehalten, so erhöht sich die Summe um den Zahlenwert der entsprechenden Taste.
Eine Auswertung erfolgt in Basic mittels der Anweisung JOY(x), wobei x für Joystick 1 eine Null und für Joystick 2 eine 1 annehmen muß, entsprechend der gewohnten Inkey-Anweisung.

Ein alternativer Programmteil könnte somit folgendermaßen aussehen:

```
80 reihe=1:spalte=1
105 eingabe=JOY(0)
115 IF eingabe=1 THEN reihe=reihe-1
125 IF eingabe=2 THEN reihe=reihe+1
135 IF eingabe=8 THEN spalte=spalte+1
145 IF eingabe=4 THEN spalte=spalte-1
155 IF eingabe=5 THEN reihe=reihe-1:s
palte=spalte-1
165 IF eingabe=9 THEN reihe=reihe-1:s
palte=spalte+1
175 IF eingabe=10 THEN reihe=reihe+1:s
palte=spalte+1
185 IF eingabe=6 THEN reihe=reihe+1:s
palte=spalte-1
```

Damit wir bei der Schaffung unserer neuen Zeichen auch jeweils wissen, welche Reihe und Spalte gerade adressiert wird, wollen wir an der entsprechenden Position ein Sternchen ausgeben lassen.
Dieses muß anschließend aber wieder gelöscht werden, ansonsten hätten wir bald ein Rechteck voller Sternchen.
Es soll also blinken, wozu wir in diesem Fall allerdings keine spezielle Routine benötigen, sondern das Programm einfach wieder von vorn beginnen lassen können:

```
225 LOCATE spalte,reihe:PRINT"*";
270 GOTO 100
```

Das Setzen eines Punktes des neuen Zeichens soll durch einen Druck auf die ENTER-Taste, welche der Computer unter dem CHR-Code 13 kennt, geschehen und uns durch das Grafikzeichen 143, einem soliden Block, angezeigt werden.
Ein falsch gesetzter Punkt soll entsprechend durch Betätigung der Leer-Taste wieder aus der Matrix entfernt werden:

```
230 IF eingabe$=CHR$(13) THEN buchstabe$
(reihe,spalte)=CHR$(143)
240 IF eingabe$=CHR$(13) THEN buchstabe$
(reihe,spalte)=CHR$(144)
250 IF eingabe$="D" THEN GOTO 300
```

Zeile 250 sorgt schließlich dafür, daß nach Eingabe eines großen 'D' ein noch anzugebendes Zeichen neu definiert wird:

```
300 LOCATE 1,1:FOR reihe=1 TO 8
310 FOR spalte=1 TO 8
320 IF buchstabe$(reihe,spalte)=CHR$(143)
 THEN bit$(reihe)=bit$(reihe)+"1" ELSE bi
t$(reihe)=bit$(reihe)+"0"
330 NEXT spalte
340 PRINT bit$(reihe):bit$(reihe)="&X"+bi
t$(reihe):code(reihe)=VAL(bit$(reihe))
360 NEXT reihe
370 INPUT"Welches Zeichen aendern (chr$)
";z
380 SYMBOL z,code(1),code(2),code(3),code
(4),code(5),code(6),code(7),code(8)
```

```
390 LOCATE 9,30:PRINT CHR$(a):GOTO 20
```

Reihe für Reihe wird das auf dem Bildschirm ausgedruckte Bild in eine Binärzahl umgesetzt (320).
Der so entstandene bit-String einer jeden Reihe wird dann als Dezimalzahl innerhalb einer Codetabelle gespeichert und nachdem der Anwender sich entschieden hat, welches Zeichen umdefiniert werden soll, wird die Manipulation vorgenommen (380) und das neue Symbol zur Kontrolle an Bildschirmposition 9,30 ausgedruckt.

STRING$

Wenn Sie sich nun fragen, wozu das Ganze gut sein soll, so kann die Antwort nur darauf hinweisen, daß Ihnen durch diese Technik die Möglichkeit zu extrem schnell aufzubauenden Bildern wie auch zu animierten Grafiken und Telespielen gegeben ist.
Denn so wie eine Text- oder Dateiverwaltung Buchstaben zu Worten zusammensetzen kann, können Sie auch Grafikstrings konstruieren:

```
10 MODE 1
20 Z$=STRING$(240,238)
30 FOR I%=1 TO 4
40 PRINT Z$;
50 NEXT I%
60 PRINT STRING$(40,238);
70 GOTO 70
```

Schneller werden Sie solch eine Grafik wohl kaum auf den Bildschirm bekommen !

Natürlich muß ein solcher String durchaus nicht nur aus einem Symbol bestehen, sondern kann entsprechend dem Verwendungszweck gestaltet sein:

```
10 MODE 0
20 M1$=CHR$(248)
30 M2$=CHR$(249)
40 M3$=CHR$(250)
50 M4$=CHR$(251)
60 MAENNER$=M1$+M2$+M3$+M4$
70 CLS
80 PRINT MAENNER$
```

```
90 PRINT
```

Der größte Vorteil dieser Grafik-Zeichenketten ist dann der, daß ihnen neben dem ganzen Spektrum der zur Verfügung stehenden String-Befehle des Locomotive Basic auch sämtliche Adressierungsarten des Bildschirmes offenstehen, d.h. diese Zeichengruppen können mit LOCATE an eine bestimmte Position des Bildes gebracht werden, oder aber auch nach Eingabe von TAG mittels des Grafikcursors an jeder der 128 000 möglichen Positionen plaziert werden.

```
10 MODE 0
20 M1$=CHR$(248)
30 M2$=CHR$(249)
40 M3$=CHR$(250)
50 M4$=CHR$(251)
60 MAENNER$=M1$+M2$+M3$+M4$
70 CLS
80 SPALTE=1:ZEILE=1:STIFT=1
90 PEN STIFT
100 LOCATE SPALTE,ZEILE
110 PRINT MAENNER$
120 ZEILE=ZEILE+1
130 IF ZEILE<25 THEN 100 ELSE ZEILE=1
140 SPALTE=SPALTE+4
150 STIFT=STIFT+1
160 IF STIFT=16 THEN STIFT=1
170 PEN STIFT
180 IF SPALTE>20 THEN SPALTE=1
190 GOTO 100
```

Steuerzeichen

Doch sind die vielfältigen Möglichkeiten der Grafikzeichen damit bei weitem noch nicht erschöpft.
Vielleicht ist es Ihnen aufgefallen, daß der Wert im Zusammenhang mit der Anweisung SYMBOL AFTER immer mindestens 32 beträgt, daß also nur die Zeichen mit den Codenummern von 32 bis 255 geändert werden können.
Dies liegt daran, daß den ersten 32 Zeichen eine feste Funktion zur Ansteuerung der Hardware zugeordnet ist, und diese kann nicht nach Belieben geändert werden, wenn der Rechner noch ordnungsgemäß funktionieren soll.
Doch bedeutet das nicht, daß wir auf diese Steuerzeichen nicht zugreifen könnten.

Beachten Sie den Unterschied des folgenden Programmes zum vorhergehenden, und Sie sehen, daß durch die CHR-Codes beispielsweise die Betriebsart gewählt, der Bildschirm gelöscht und auch die Farben gewählt werden können:

```
10 PRINT CHR$(4);CHR$(0)
20 M1$=CHR$(248)
30 M2$=CHR$(249)
40 M3$=CHR$(250)
50 M4$=CHR$(251)
60 MAENNER$=M1$+M2$+M3$+M4$
70 PRINT CHR$(12)
80 SPALTE=1:ZEILE=1:STIFT=1
90 PRINT CHR$(15);CHR$(STIFT)
100 LOCATE SPALTE,ZEILE
110 PRINT MAENNER$
120 ZEILE=ZEILE+1
130 IF ZEILE<25 THEN 100 ELSE ZEILE=1
140 SPALTE=SPALTE+4
150 STIFT=STIFT+1
```

```
160 IF STIFT=16 THEN STIFT=1
170 PRINT CHR$(15);CHR$(STIFT)
180 IF SPALTE>20 THEN SPALTE=1
190 GOTO 100
```

Doch das ist immer noch nicht alles, was Ihr Schneider kann. So ist es auch möglich, den Cursor so an den verschiedensten Stellen des Bildschirmes zu positionieren.

Auskunft über die beim Schneider CPC zur Verfügung stehenden Steuerzeichen gibt die folgende Aufstellung.
Im praktischen Gebrauch bieten Ihnen die ersten 32 Zeichen sogar eine Doppelfunktion an, denn die Steuerzeichen werden auch durch ein Grafikzeichen repräsentiert.
Allerdings ist es erforderlich, vor der Abbildung eines dieser Zeichen im Direktmodus zunächst die Control-Taste zu drücken, oder innerhalb eines Programmes erst ein PRINT CHR$(2); durchzuführen.

Da die grafischen Äquivalente dieser Steuerzeichen nicht im Anhang III des Schneider Handbuches abgebildet wurden, wollen wir Sie Ihnen an dieser Stelle zunächst zeigen, anschließend wollen wir auf die Steuerfunktionen dieser Codes eingehen.

3800	FF	3808	FF
3801	C3	3809	C0
3802	C3	380A	C0
3803	C3	380B	C0
3804	C3	380C	C0
3805	C3	380D	C0
3806	C3	380E	C0
3807	FF	380F	C0

3810	18	3818	03
3811	18	3819	03
3812	18	381A	03
3813	18	381B	03
3814	18	381C	03
3815	18	381D	03
3816	18	381E	03
3817	FF	381F	FF

3820	0C	3828	FF
3821	18	3829	C3
3822	30	382A	E7
3823	7E	382B	DB
3824	0C	382C	DB
3825	18	382D	E7
3826	30	382E	C3
3827	00	382F	FF
3830	00	3838	3C
3831	01	3839	66
3832	03	383A	C3
3833	06	383B	C3
3834	CC	383C	FF
3835	78	383D	24
3836	30	383E	E7
3837	00	383F	00
3840	00	3848	00
3841	00	3849	00
3842	30	384A	0C
3843	60	384B	06
3844	FF	384C	FF
3845	60	384D	06
3846	30	384E	0C
3847	00	384F	00
3850	18	3858	18
3851	18	3859	3C
3852	18	385A	7E
3853	18	385B	DB
3854	DB	385C	18
3855	7E	385D	18
3856	3C	385E	18
3857	18	385F	18
3860	18	3868	00
3861	5A	3869	03
3862	3C	386A	33
3863	99	386B	63
3864	DB	386C	FE
3865	7E	386D	60
3866	3C	386E	30
3867	18	386F	00
3870	3C	3878	3C
3871	66	3879	66
3872	FF	387A	C3
3873	DB	387B	DB
3874	DB	387C	DB
3875	FF	387D	C3
3876	66	387E	66
3877	3C	387F	3C
3880	FF	3888	3C
3881	C3	3889	7E
3882	C3	388A	DB
3883	FF	388B	DB
3884	C3	388C	DF
3885	C3	388D	C3
3886	C3	388E	66
3887	FF	388F	3C

3890	3C
3891	66
3892	C3
3893	DF
3894	DB
3895	DB
3896	7E
3897	3C

3898	3C
3899	66
389A	C3
389B	FB
389C	DB
389D	DB
389E	7E
389F	3C

38A0	3C
38A1	7E
38A2	DB
38A3	DB
38A4	FB
38A5	C3
38A6	66
38A7	3C

38A8	00
38A9	01
38AA	33
38AB	1E
38AC	CE
38AD	7B
38AE	31
38AF	00

38B0	7E
38B1	66
38B2	66
38B3	66
38B4	66
38B5	66
38B6	66
38B7	E7

38B8	03
38B9	03
38BA	03
38BB	FF
38BC	03
38BD	03
38BE	03
38BF	00

38C0	FF
38C1	66
38C2	3C
38C3	18
38C4	18
38C5	3C
38C6	66
38C7	FF

38C8	18
38C9	18
38CA	3C
38CB	3C
38CC	3C
38CD	3C
38CE	18
38CF	18

38D0	3C
38D1	66
38D2	66
38D3	30
38D4	18
38D5	00
38D6	18
38D7	00

38D8	3C
38D9	66
38DA	C3
38DB	FF
38DC	C3
38DD	C3
38DE	66
38DF	3C

38E0	FF
38E1	DB
38E2	DB
38E3	DB
38E4	FB
38E5	C3
38E6	C3
38E7	FF

38E8	FF
38E9	C3
38EA	C3
38EB	FB
38EC	DB
38ED	DB
38EE	DB
38EF	FF

38F0	FF
38F1	C3
38F2	C3
38F3	DF
38F4	DB
38F5	DB
38F6	DB
38F7	FF

38F8	FF
38F9	DB
38FA	DB
38FB	DB
38FC	DF
38FD	C3
38FE	C3
38FF	FF

Die Steuerzeichen

Die durch die Codes von 1 bis 31 belegten Steuerfunktionen des Schneider CPC können alle mittels einer PRINT-Anweisung zur Ausführung gebracht werden.

Beispiel: PRINT CHR$(7) läßt einen einzelnen Ton erklingen.

Komplexere Arbeitsabläufe erfordern weitere Angaben zu der Anweisung. Lassen Sie in diesem Falle zunächst ein Semikolon folgen und anschließend die CHR$-Codes der Parameter.

Beispiel: PRINT CHR$(1);CHR$(1) gibt ein Rechteck auf den Bildschirm aus.

Die Codes im einzelnen:

CHR$(0)

ist nicht definiert.

CHR$(1)

veranlaßt eine grafische Ausgabe des nachstehenden Steuerzeichens. So ergibt die Befehlsfolge PRINT CHR$(1);CHR$(0) ein Rechteck an der aktuellen Cursorposition.

CHR$(2)

Nach dieser Anweisung wird der Textcursor nicht mehr abgebildet.

CHR$(3)

schaltet den Textcursor wieder ein.

CHR$(4)

Der Wert der nachfolgenden PRINT CHR$() Anweisung bestimmt die Betriebsart des CPC. Dabei wird eine Auswertung Modulo 4 vorgenommen, das heißt nicht der eigentliche Wert gilt, sondern der Rest bei einer Division durch vier.
PRINT CHR$(4);CHR$(5) ist somit gleichwertig mit PRINT CHR$(4);CHR$(1); beide Anweisungen schalten Mode 1 ein.

CHR$(5)

entspricht in der Funktionsweise PRINT CHR$(2), nur wird das Zeichen hier an der aktuellen Position des Grafikcursors ausgegeben.

CHR$(6)

schaltet den Textbildschirm ein.

CHR$(7)

entspicht dem ASCII-Bell; läßt einen Pieps ertönen und leert die Tonwarteschlangen

CHR$(8)

bewegt den Cursor um ein Zeichen zurück

CHR$(9)

bewegt den Cursor um ein Zeichen vor

CHR$(10)

bewegt den Cursor eine Zeile tiefer

CHR$(11)

bewegt den Cursor um eine Zeile nach oben

CHR$(12)

entspricht dem Befehl CLS.

CHR$(13)

entspricht dem Drücken der ENTER-Taste.

CHR$(14)

interpretiert den nachfolgenden Codewert MODULO 16 als PAPER-Anweisung.

CHR$(15)

interpretiert den nachfolgenden Codewert als PEN-Anweisung:
PRINT CHR$(15);CHR$(2) entspricht PEN 2.
Es gilt das Ergebnis der Restedivision MOD 16.

CHR$(16)

löscht das Zeichen unter dem Cursor.

CHR$(17)

löscht die aktuelle Zeile bis zum Cursor.

CHR$(18)

löscht die aktuelle Zeile vom Cursor bis zum Zeilenende.

CHR$(19)

löscht alle Bildschirmpositionen von 1,1 bis zum Cursor.

CHR$(20)

löscht vom Cursor bis zum Bildschirmende.

CHR$(21)

schaltet den Textbildschirm ab.

CHR$(22)

Der Restwert einer Division des folgenden Codewertes durch zwei dient zum Ein- bzw. Ausschalten des Transparentmodus.

PRINT CHR$(22);CHR$(0) schaltet den Transparentmodus aus, und PRINT CHR$(22);CHR$(1) schaltet den Transparentmodus ein.

CHR$(23)

Das Ergebnis des nachfolgenden Wertes MOD 4 entscheidet über den Grafikschreibmodus.

Wird nach CHR$(22) ein CHR$(0) an den Bildschirm gesandt, ist der normale Betriebszustand eingeschaltet, das heißt, alle Punkte, welche bei der Ausführung des nächsten Grafikbefehles gesetzt werden müßten, werden gesetzt.

Die anderen drei Möglichkeiten setzen die neu zu setzenden Punkte zunächst in Relation zu bereits an den betreffenden Koordinaten gesetzten Punkten.

So werden nach PRINT CHR$(22);CHR$(1) nur noch Punkte gesetzt, welche entweder schon gesetzt waren oder noch gesetzt werden;
folgte ein CHR$(2), leuchten nur diejenigen Punkte auf, welche in beiden Teilbildern gesetzt waren.

CHR$(24)

vertauscht die PAPER und PEN Farben.

CHR$(25)

entspricht dem Symbol Kommando. Wie auch von Basic gewohnt, müssen nach PRINT CHR$(25); neun Parameter folgen, welche das zu ändernde Zeichen und die zu setzenden Punkte angeben.

CHR$(26)

entspricht der Basic-Anweisung WINDOW. Es müssen vier Parameter zur Festlegung der Eckpunkte folgen.

CHR$(28)

entspricht dem INK-Kommando; es müssen drei Parameter für Stiftnummer, Farbe1 und Farbe2 folgen.

CHR$(29)

entspricht der Anweisung BORDER; Angaben zur Farbe1 und Farbe2 müssen folgen.

CHR$(30)

setzt den Cursor in die linke obere Ecke des aktivierten Fensters.

CHR$(31)

ist vollwertiger Ersatz für das LOCATE-Kommando. Wie üblich folgt nach der Anweisung zunächst der Spalten- und dann der Zeilenwert.

Zugegeben, nicht alle Steuercodes verdienen unsere Beachtung, werfen einige doch die Frage auf, warum einfach (als Basic-Anweisung), wenn es auch kompliziert geht, doch verdienen insbesondere die Cursorsteuerbefehle wie auch die Codes zur Farb- und Grafikmoduseinstellung unsere Aufmerksamkeit.

Denn bei geschicktem Gebrauch dieser 'Zeichen' stehen einem Schneider-Anwender weitaus mehr Möglichkeiten offen, als einem Commodore-Fan mit seinen Sprites.

Action-Spiele

mit ihren oft fantastischen Grafiken sind für viele Leute das Größte, und so beurteilen diese Menschen ihren Computer häufig eher nach den auf ihm laufenden Spielen als nach der Handhabung ihrer Programmierung.

Für den gewerbsmäßigen Einsatz in Spielhallen werden meist spezialisierte Microcomputer eingesetzt, die häufig neben Hardwareerweiterungen auch über besondere Software verfügen. Auf der technischen Seite können dies eine Vielzahl von Sprites sein, auf die rasch zugegriffen werden kann, wie auch eine extrem hochauflösende, farbkräftige Bildröhre.

Auf der programmtechnischen Seite sah es früher so aus, daß diese Spiele in reiner Maschinensprache geschrieben wurden, um einen raschen Spielverlauf zu gewährleisten.
Heute dagegen bedient man sich auch auf diesem Gebiet höherer Programmiersprachen, um dadurch die Entwicklungzeit möglichst kurz und die (Lohn-) Kosten entsprechend niedrig zu halten.
Meistens bedient man sich dabei der Sprache FORTH, die im Bereich der Astronomie entwickelt wurde und die es erlaubt, ihren eigenen Wortschatz um beliebig viele Schlüsselworte zu erweitern.
So könnte es in einer GAMEFORTH-Version Befehle wie JOYMOVE oder EXPLODE geben, deren einfacher Einbau in das Forth-Programm dann die Bewegung einer Figur in Abhängigkeit des Joysticks durchführt oder eine Explosion in Bild und Ton nachahmt.

Heimcomputer sind jedoch Allzweckgeräte. Da aber den Herstellern das Interesse der Kundschaft an diesen Actionspielen durchaus bewußt geworden ist, lassen sich auch diesen Geräten die eine oder andere spezielle Funktion entlocken.

So auch der Schneider CPC. Zwar kennt er keine Sprites, doch kann auch er mit einer Menge Unterstützung bei der Spieleprogrammierung aufwarten.

Die dazu notwendigen Techniken zur Grafikerzeugung und Animation sollen nun Thema der folgenden Seiten sein.

Grafik-Zeichenketten

können in fast beliebigen Mengen erzeugt werden. Allein der zur Verfügung stehende Speicherplatz könnte einem überaus kreativem Spiele-Programmierer ein Hemmschuh sein.
Doch wenn dieser Fall eintritt, wird es in dem erzeugten Programm nur so von Raketen und Schiffen, Monstern, Menschen und dergleichen mehr wimmeln.

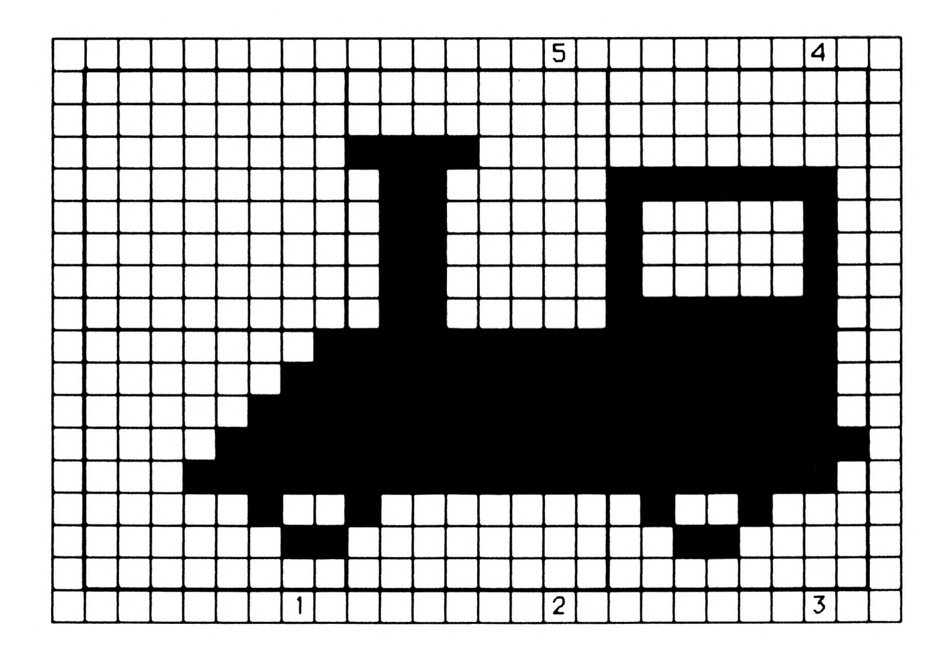
Da diese einzelnen Elemente zum einen gut zu sehen sein und außerdem über einige Details verfügen sollen, müssen sie auch eine entsprechende Größe vorweisen.
Einfach Mode 0 wählen und dann ein Zeichen umändern, würde zwar auf den ersten Blick genügen, doch legt der zweite Blick die Schwachstellen bloß:

- denn erstens sind die Bewegungsabläufe in dieser Betriebsart immer recht holperig und

- zweitens kann jede Figur in höchstens einer Farbe gezeichnet werden.

Somit kommt als Betriebsart nur Mode 1 in Frage, denn dann ist die Auflösung einigermaßen, und außerdem stehen uns immerhin noch vier Farben insgesamt zur Verfügung.
Die Spielfiguren müssen dann aus mehreren einzelnen Zeichen zusammengesetzt werden, genau in der Art, wie wir es schon mit den Männern getan haben.

Würden wir beispielsweise eine Lokomotive benötigen, legen wir zunächst ihre ungefähre Größe, vielleicht zwei mal drei Zeichen, fest, und zeichnen ihren Umriß auf ein entsprechend markiertes Blatt kariertes Papier:



Anschließend füllen wir diesen Umriß entsprechend unseren Vorstellungen aus, und machen uns an die Redefinition einer entsprechenden Anzahl von Zeichen.
Hier sind es fünf, und die Symbolfolgen für die einzelnen Zeichen lauten:

```
1,3,7,15,31,4,3,0
255,255,255,255,255,128,0,0
254,254,254,255,254,72,48,0
0,0,0,254,130,130,130,254
0,0,240,96,96,96,96,96
```

Diese Werte können wir entweder berechnen oder uns von unserem Zeicheneditor liefern lassen.
Doch nützt es uns wenig, wenn die Zeichen nur einmal geändert werden, was wir brauchen, ist ein Programmteil, welcher korrekte Basic-Anweisungen erhält.
Warum sollte der Editor uns nicht dieses Modul liefern ?

Nur einige wenige Ergänzungen werden erforderlich, dann nimmt uns der Computer die komplette Arbeit ab.
Wir lassen einfach sämtliche Codewerte für ein jedes Zeichen in einem weiteren Feld abspeichern, und wenn wir mit der creativen Arbeit fertig sind, lassen wir diese Daten, versehen mit der Anweisung SYMBOL, als File auf unseren Datenträger schreiben.
Dieses können wir dann später entweder ganz normal laden, oder aber mittels der Anweisung MERGE an ein im Speicher befindliches Programm anfügen.

Aus diesem Grunde folgt nun zunächst das komplette Listing des Zeichengenerators ZPRO.

```
5 MODE 1:PAPER 0:INK 0,12:BORDER 12:PEN
1:INK 1,0:SYMBOL AFTER 33
10 DEFINT a-z:DIM buchstabe$(8,8):anz=0
20 CLS:FOR reihe=1 TO 8
30 FOR spalte=1 TO 8
40 buchstabe$(reihe,spalte)=CHR$(144)
50 NEXT spalte
60 bit$(reihe)=""
70 NEXT reihe
80 GOSUB 400:GOSUB 500
90 reihe=1:spalte=1
100 eingabe$=INKEY$:eingabe=JOY(0)
110 IF eingabe$="8" THEN reihe=reihe-1
115 IF eingabe=1 THEN reihe=reihe-1
120 IF eingabe$="2" THEN reihe=reihe+1
125 IF eingabe=2 THEN reihe=reihe+1
130 IF eingabe$="6" THEN spalte=spalte+1
135 IF eingabe=8 THEN spalte=spalte+1

140 IF eingabe$="4" THEN spalte=spalte-1
145 IF eingabe=4 THEN spalte=spalte-1
150 IF eingabe$="7" THEN reihe=reihe-1:s
palte=spalte-1
155 IF eingabe=5 THEN reihe=reihe-1:spal
te=spalte-1
160 IF eingabe$="9" THEN reihe=reihe-1:s
palte=spalte+1
165 IF eingabe=9 THEN reihe=reihe-1:spal
te=spalte+1
170 IF eingabe$="3" THEN reihe=reihe+1:s
palte=spalte+1
175 IF eingabe=10 THEN reihe=reihe+1:spa
lte=spalte+1
180 IF eingabe$="1" THEN reihe=reihe+1:s
palte=spalte-1
185 IF eingabe=6 THEN reihe=reihe+1:spal
te=spalte-1
```

```
190 IF reihe<1 THEN reihe=1:PRINT CHR$(7
)
200 IF reihe>8 THEN reihe=8:PRINT CHR$(7
)
210 IF spalte<1 THEN spalte=1:PRINT CHR$
(7)
220 IF spalte>8 THEN spalte=8:PRINT CHR$
(7)
225 LOCATE spalte,reihe:PRINT"*";
230 IF eingabe$=CHR$(13) THEN buchstabe$
(reihe,spalte)=CHR$(143)
235 IF eingabe=17 THEN buchstabe$(reihe,
spalte)=CHR$(143)
240 IF eingabe$=CHR$(32) THEN buchstabe$
(reihe,spalte)=CHR$(144)
245 IF eingabe=18 THEN buchstabe$(reihe,
spalte)=CHR$(144)
250 IF eingabe$="P"THEN GOTO 300
251 IF eingabe$="D" THEN GOTO 600
255 LOCATE 9,8:PRINT reihe,spalte
260 GOSUB 500
270 GOTO 100
300 LOCATE 1,1:FOR reihe=1 TO 8
310 FOR spalte=1 TO 8
320 IF buchstabe$(reihe,spalte)=CHR$(143
) THEN bit$(reihe)=bit$(reihe)+"1" ELSE
bit$(reihe)=bit$(reihe)+"0"
330 NEXT spalte
340 PRINT bit$(reihe):bit$(reihe)="&x"+b
it$(reihe):code(reihe)=VAL(bit$(reihe))
360 NEXT reihe
370 INPUT"Welches Zeichen aendern (chr$)
 ";z
380 SYMBOL z,code(1),code(2),code(3),cod
e(4),code(5),code(6),code(7),code(8)
385 anz=anz+1:zalt(anz)=z:FOR i=1 TO 8:z
code(anz,i)=code(i):NEXT i
```

```
390 LOCATE 9,30:PRINT CHR$(a):GOTO 20
400 WINDOW#1,10,40,1,10
410 PRINT#1,"* * * *    Z - PRO    * * *
 *"
420 PRINT#1:PRINT#1,"Cursorbewegung:   1
0'er Block":PRINT#1,"                  J
oystick"
430 PRINT#1,"Pixel setzen  :   <ENTER>

440 PRINT#1:PRINT#1,"Neudefininieren:  S
HIFT p"
450 RETURN
499 REM matrix ausgeben
500 LOCATE 1,1:FOR re=1 TO 8
510 FOR sp=1 TO 8
520 PRINT buchstabe$(re,sp);
530 NEXT sp
540 PRINT
550 NEXT re
570 RETURN
600 WINDOW#2,1,40,10,20:CLS#2
610 PRINT#2,"Datentraeger bereitmachen !
"
620 PRINT#2,"DATA-File generieren:"
630 INPUT#2,"Name ";file$:IF LEN(file$)>
8 THEN 630
640 INPUT#2,"erste Zeilennummer ";zn
650 INPUT#2,"Schrittweite ";sw
660 file$="!"+file$
670 OPENOUT file$
680 PRINT#9,STR$(zn)+"SYMBOL AFTER 32"
690 FOR z=1 TO anz
700 PRINT#9,STR$(zn)+"SYMBOL "+STR$(zalt
(z))+","+STR$(zcode(z,1))+","+STR$(zcode
(z,2))+","+STR$(zcode(z,3))+","+STR$(zco
de(z,4))+","+STR$(zcode(z,5))+","+STR$(z
code(z,6))+","+STR$(zcode(z,7))+","+STR$
```

```
(zcode(z,8))
710 zn=zn+sw
720 NEXT z
730 CLOSEOUT
740 RUN
```

Für unser Beispiel haben wir die Zeichen 130,131,132,133 und 134 geändert, und zwar so daß sie gegenläufig zum Uhrzeigersinn die einzelnen Elemente unserer Lok beschreiben.

Um diese nun auf dem Bildschirm abzubilden, bedienen wir uns dieser Zeichen, wie auch der Cursor-Steuercodes, da unser Werk nicht in eine einzige Zeile paßt:

```
10 REM zug1
20 MODE 0
30 SYMBOL AFTER 129
40 SYMBOL  130, 1, 3, 7, 15, 31, 4, 3, 0
50 SYMBOL  131, 255, 255, 255, 255, 255,
 128, 0, 0
60 SYMBOL  132, 254, 254, 254, 255, 254,
 72, 48, 0
70 SYMBOL  133, 0, 0, 0, 254, 130, 130,
130, 254
80 SYMBOL  134, 0, 0, 240, 96, 96, 96, 9
6, 96
90 LOCATE 10,10:REM wo?
100 PRINT CHR$(130);:REM zugspitze
110 PRINT CHR$(131);:REM kessel
120 PRINT CHR$(132);:REM ende der lok
130 PRINT CHR$(11);:REM eine zeile rauf

140 PRINT STRING$(2,8);:REM zwei nach li
nks
150 PRINT CHR$(134);:REM schornstein
160 PRINT CHR$(133);:REM fuehrerhaus
170 LOCATE 1,1
```

Wenn diese Lokomotive zum Zwecke der Fortbewegung nun wieder und wieder abgebildet werden muß, dann erweist sich die Abbildung in Form einzelner Zeichen als zu aufwendig und langwierig.
Glücklicherweise jedoch kann eine Zeichenkette im Schneider-Basic bis zu 255 Zeichen lang sein, und das dürfte für alle Zwecke mehr als genug sein.

```
180 zug$=CHR$(130)+CHR$(131)+CHR$(132)+C
HR$(11)+STRING$(2,8)+CHR$(134)+CHR$(133)
190 LOCATE 3,3
200 PRINT zug$;
210 LOCATE 1,1
```

Auf diese Art und Weise können nun zu Beginn eines jeden Spielprogrammes zunächst die Zeichen neugestaltet, und daran anschließend können diese Bausteine zu den Elementen des Spieles verknüpft werden.
Dabei erweist es sich als großer Vorteil, daß der Programmierer sinngemäße Bezeichnungen verwenden kann, die das gesamte Programm überschaubar und leicht editierbar halten. Er muß nicht mit bloßen Zahlenwerten hantieren, deren Bedeutung er nach einigen Stunden selbst nicht mehr kennt.

Mehrfarbige Darstellungen

sind mit dieser Konzeption ebenfalls leicht zu realisieren. Es wurde bereits auf den Steuercode 15 hingewiesen, der einen vollwertigen Ersatz für das PEN-Kommando darstellt.

So bewirkt folgende Änderung des Programmes, daß der Rumpf der Lokomotive schwarz und die Aufbauten, also der

Schornstein und das Führerhaus, blau dargestellt werden.

Beachten Sie aber bitte, daß es sich um die PEN und nicht um eine INK-Anweisung handelt, weshalb Sie nicht vergessen dürfen, die Auswahl der Farben im Programmkopf bei der Definition der Betriebsart mit festzulegen (Zeile 20):

```
10 REM zug1
20 MODE 0:INK 1,1:INK 2,0
30 SYMBOL AFTER 129
40 SYMBOL  130, 1, 3, 7, 15, 31, 4, 3, 0
50 SYMBOL  131, 255, 255, 255, 255, 255,
 128, 0, 0
60 SYMBOL  132, 254, 254, 254, 255, 254,
 72, 48, 0
70 SYMBOL  133, 0, 0, 0, 254, 130, 130,
130, 254
80 SYMBOL  134, 0, 0, 240, 96, 96, 96, 9
6, 96
90 LOCATE 10,10:REM wo?
100 PRINT CHR$(130);:REM zugspitze
110 PRINT CHR$(131);:REM kessel
120 PRINT CHR$(132);:REM ende der lok
130 PRINT CHR$(11);:REM eine zeile rauf

140 PRINT STRING$(2,8);:REM zwei nach li
nks
150 PRINT CHR$(15);CHR$(2);:REM ab jetzt
 pen2
160 PRINT CHR$(134);:REM schornstein
170 PRINT CHR$(133);:REM fuehrerhaus
180 PRINT CHR$(15);CHR$(1);:REM und wied
er pen1
190 LOCATE 1,1
```

Selbstverständlich können diese Aktionen ebenfalls in die Grafik-Zeichenketten eingebaut werden:

```
200 stift1$=CHR$(15)+CHR$(1)
210 stift2$=CHR$(15)+CHR$(2)
220 zug$=CHR$(130)+CHR$(131)+CHR$(132)+C
HR$(11)+STRING$(2,8)+stift2$+CHR$(134)+s
tift1$+CHR$(133)
230 LOCATE 4,4:PRINT zug$
```

Diese Zeichenketten können dann mit sämtlichen String-Befehlen manipuliert werden, daß heißt, Sie können Teilstrings entnehmen oder auch einzelne Elemente austauschen - machbar ist alles !

Animation

Der naturgetreue Eindruck, den Bewegungsabläufe bei der Projektion eines Filmes hinterlassen, entsteht durch die Vielzahl der Einzelbilder, welche dem menschlichen Auge dargeboten werden.
Betrachtet man die 25 Bilder eines Fernseh- oder die 18 Bilder eines 8 mm Filmes, welche in jeweils einer Sekunde gezeigt werden, nebeneinander, so stellt man fest, daß jede Bewegung innerhalb dieser Sekunde in entsprechend viele Teilstrecken zerlegt wurde.
Es bleibt also bei einem ruckartigen Versetzen des Objektes, nur ist das Auge nicht in der Lage, mehr als vierzehn einzelne Positionen innerhalb einer Sekunde zu erkennen.

Die Trägheit des Auges ist somit zuständig für das Erlebnis Film, doch gibt es Situationen, in denen es sich nicht täuschen läßt:
Denn wenn die Differenz zwischen den einzelnen Positionen des dargestellten Objektes zu groß wird, dann bewegt es sich trotz großer Bildfrequenz ruckartig voran.
Im Film wird diese Erscheinung als Shutter bezeichnet, und wegen dieses Effektes werden auch nicht allzu oft Einstellungen gedreht, bei denen sich das Objekt geradlinig von der einen Seite des Bildschirmes zur anderen bewegt.
Da ist es den Regisseuren weitaus lieber, wenn der verfolgte Wagen aus der Tiefe des Raumes auftaucht, und dann nach einem Schwenk wieder dort verschwindet.

Auch wir werden uns diese Technik der Einzelbilder zunutze machen, und würden wir diese Vorgehensweise konsequent genug bei unseren computeranimierten Bildern anwenden, so wäre auch das Ergebnis auf dem Datenmonitor von hervorragender Trickfilmqualität.

Doch heißt das nicht mehr und nicht weniger, als daß wir mindestens sechzehn mal in der Sekunde das gesamte Bild aufbauen müssen, und diese Fleißarbeit im Punktesetzen (16 x 640 x 200) wäre selbst für ein Programm in Maschinensprache zu viel.

Doch haben wir es ja nicht nötig, immer gleich das ganze Bild neu abzubilden, sondern können uns mit der Darstellung des bewegten Objektes begnügen.

Wir werden einen Punkt setzen, diesen löschen, die neue Position berechnen, den Punkt setzen, ihn löschen, die neue Position berechnen ...

```
200 FOR zeile=1 TO 24
210 FOR spalte=1 TO 40
220 LOCATE spalte,zeile
230 PRINT " *";
240 NEXT spalte
250 NEXT zeile
```

Nach dem Start mit RUN 200 wird der Stern sich Schritt für Schritt von oben links nach unten rechts bewegen.
Denn da wir zuerst immer ein Leerzeichen und erst dann die Bildfigur ausgeben, andererseits die neue Bildposition aber immer nur um einen Schritt weiterbewegen, wird das alte Bild automatisch gelöscht.

Auch rückwärts geht's ganz gut, wie der Versuch mit PRINT "* " und abwärtszählenden Schleifen beweist.
Doch ergeben sich bei dieser Verfahrensweise Schwierigkeiten an den Rändern, wenn nämlich bereits die nächste Zeile angesprochen wird.

Ein Ausweg aus dieser Situation ist die Speicherung der letzten Druckposition, denn dann kann das alte Bild vor der Ausgabe des neuen zunächst gelöscht werden:

```
199 salt=1:zalt=1
200 FOR zeile=1 TO 24
210 FOR spalte=1 TO 40
220 LOCATE salt,zalt
230 PRINT " ";
240 LOCATE spalte,zeile
250 PRINT"*";
260 salt=spalte:zalt=zeile
270 NEXT spalte
280 NEXT zeile
```

Allerdings dürfen Sie bei diesem Verfahren nicht vergessen, die ersten Löschwerte 'von Hand' einzugeben wie es hier in Zeile 199 geschehen ist.
Ansonsten funktioniert alles recht ordentlich, doch geht es sogar noch etwas besser.

Denn der CPC verfügt über eine Betriebssystemroutine, welche den Rücklauf des Bildröhrenstrahles abwartet; ein Bild wird erst dann gezeigt, wenn es ganz aufgebaut ist.
Angesprochen werden kann dieses Unterprogramm über den Sprungvektor BD19; selbstverständlich auch von BASIC aus.

Und so ergänzen wir

```
245 CALL &BD19
```

und haben einen in perfekter Weise über den Bildschirm gleitenden Stern.
Doch kann es stattdessen auch unser Zug oder jedes andere Objekt sein, wie das Beispiel der nächsten Seite beweist.

```
10 REM zug2
20 MODE 2:DEFINT a-z
30 SYMBOL AFTER 129
40 SYMBOL  130, 1, 3, 7, 15, 31, 4, 3, 0
50 SYMBOL  131, 255, 255, 255, 255, 255,
 128, 0, 0
60 SYMBOL  132, 254, 254, 254, 255, 254,
 72, 48, 0
70 SYMBOL  133, 0, 0, 0, 254, 130, 130,
130, 254
80 SYMBOL  134, 0, 0, 240, 96, 96, 96, 9
6, 96
90 zug$=CHR$(130)+CHR$(131)+CHR$(132)+CH
R$(11)+STRING$(2,8)+CHR$(134)+CHR$(133)
100 zugloesch$="    "+CHR$(11)+STRING$(2,
8)+"  "
110 ORIGIN 0,47:DRAW 640,0
120 salt=73:zalt=22
130 zeile=22
140 PRINT CHR$(2)
150 FOR spalte=73 TO 1 STEP -1
160 LOCATE salt,zalt
170 CALL &BD19
180 PRINT zugloesch$;
190 LOCATE spalte,zeile
200 CALL &BD19
210 PRINT zug$;
220 salt=spalte
230 NEXT spalte
240 salt=1:GOTO 150
```

Unser Interesse an diesem Listing gilt der Definition einer Zeichenkette, in der Leerzeichen so angeordnet sind wie die einzelnen Bausteine des gezeichneten Objektes.
Schließlich kann ein Leerzeichen allein nicht den viel längeren zug$ überdecken.

Der nächste Schritt sieht dann so aus, daß wir die Positionsangabe mittels Locate aufgeben und uns der Möglichkeit zuwenden, Zeichenketten mit dem Grafikcursor an jeder beliebige Stelle des 640 mal 200 Punkte großen Bildschirmes auszugeben.

Statt LOCATE verwenden wir MOVE, außerdem passen wir die Schleife den neuen Gegebenheiten an:
Wenn ein Zeichen aus 8 mal 8 Pixels besteht, dann dürfte eine Schrittweite von zehn gerade richtig sein.

Allerdings, einen Nachteil müssen wir hinnehmen, denn die Cursorsteuerbefehle gelten nicht für den Grafikcursor.
Und so wird aus unserer Dampflok dann auch eher ein modernerer Triebwagen:

```
10 REM zug3
20 MODE 2:DEFINT a-z
30 SYMBOL AFTER 129
40 SYMBOL  130, 1, 3, 7, 15, 31, 4, 3, 0
50 SYMBOL  131, 255, 255, 255, 255, 255,
 128, 0, 0
60 SYMBOL  132, 254, 254, 254, 255, 254,
 72, 48, 0
70 MODE 2
80 ORIGIN 1,1
90 PLOT 1,35: DRAW 640,35
100 a$=CHR$(130)+CHR$(131)+CHR$(132)
110 TAG
120 FOR x=601 TO 1 STEP -10
130 MOVE xalt,50
140 CALL &BD19
150 PRINT"    ";
160 MOVE x,50
170 CALL &BD19
180 PRINT a$;
190 xalt=x
200 NEXT x
210 GOTO 120
```

Sprites - aber warum denn ?

Diese Frage darf jeder Besitzer eines CPC 464 demjenigen stellen, der glaubt, daß ihm mit der Handvoll Sprites seines Computers das Nonplusultra der Grafikprogrammierung zur Verfügung steht.
Sicher, Sprites lassen sich einfach steuern und zerstören vor allem den Hintergrund nicht, doch wer sagt, daß die von uns benutzten Zeichenketten das tun müssen.
Denn die Konstrukteure des CPC haben auch diese Möglichkeit berücksichtigt, nur gehört zur Nutzung dieses Angebotes etwas mehr Computererfahrung als sie ein Neuling hat und als sie das Handbuch vermittelt.

CHR$(23) heißt das Zauberwort, und schon kann auch unser Zug an jedem Bahnhof vorbeifahren, ohne daß wir diesen anschließend neu zeichnen müssen.

Doch wie soll das funktionieren ?

Ganz einfach, je nach Erfordernis der Lage werden die alten und die neuen Grafikpunkte logisch miteinander verknüft - entweder mit OR, mit AND oder gar mit XOR.

Bedauerlicherweise werden diese Worte den meisten Basic-Programmierern wohl kaum etwas sagen, denn sie können sich, im Gegensatz zu denen, die der Maschinensprache huldigen, fast immer um diese etwas komplizierteren Anweisungen herumdrücken.
OR, XOR und AND werden zum einen benötigt zur Formulierung komplexer Bedingungen, die aber auch durch mehrere IF-Statements ersetzt werden können, oder sie dienen dazu, um Bits in einem Byte zu setzen, zu löschen oder zu ermitteln.

Eingeführt werden diese logischen Operatoren meistens mit Wahrheitstabellen, denn daraus läßt sich ihre Funktionsweise am ehesten ablesen.

So gilt für eine UND-Verknüpfung, daß das Ergebnis nur dann richtig ist, wenn beide Voraussetzungen richtig waren:

0	AND	0	>	0
0	AND	1	>	0
1	AND	0	>	0
1	AND	1	>	1

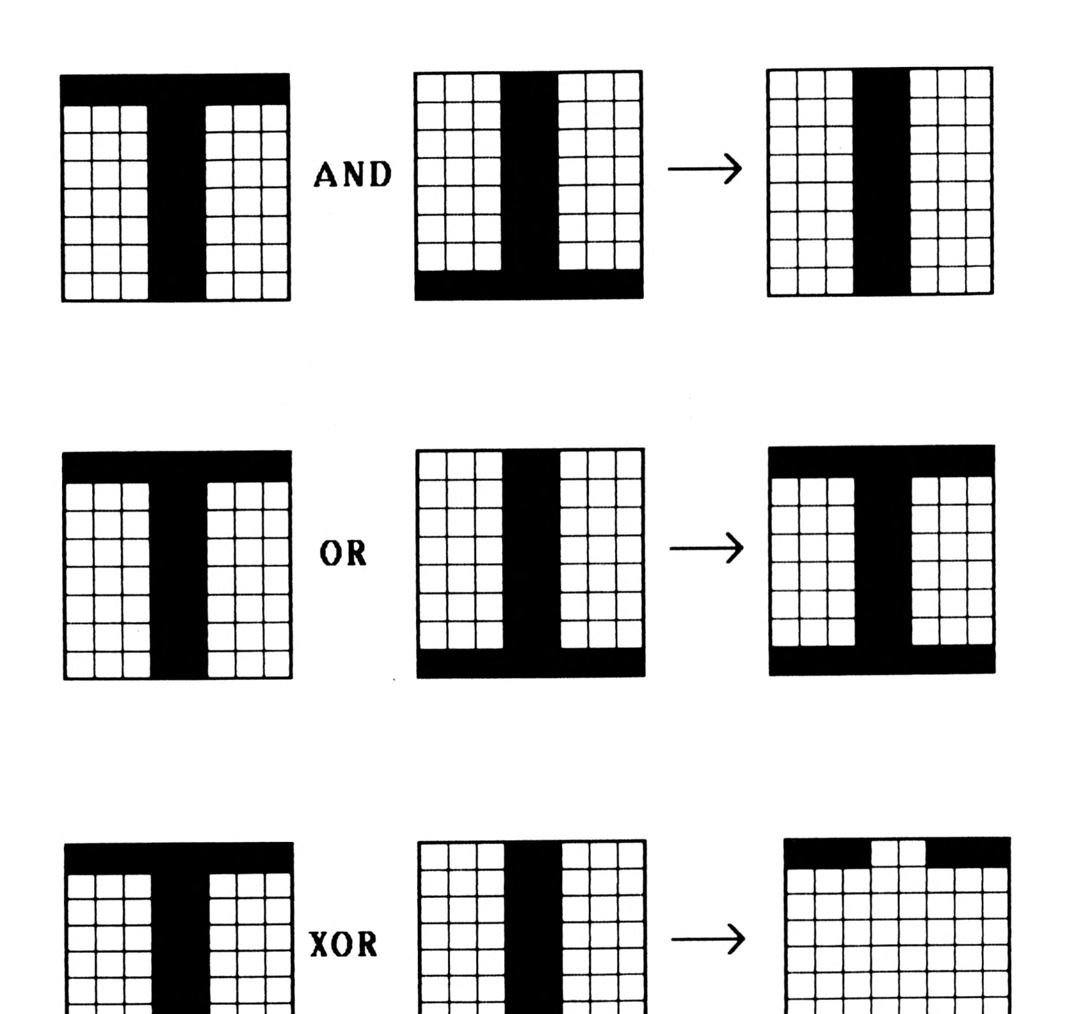
Das Ergebnis einer ODER-Verknüpfung führt dann zu einem 'wahr', wenn mindestens ein Wert wahr war:

0	OR	0	>	0
0	OR	1	>	1
1	OR	0	>	1
1	OR	1	>	1

XOR schließlich, wird übersetzt mit 'ausschließendes Oder', und bedeutet, daß nicht nur mindestens ein Wert wahr sein muß, sondern daß höchstens ein Wert wahr sein darf:

0	XOR	0	>	0
0	XOR	1	>	1
1	XOR	0	>	1
1	XOR	1	>	0

Wenn Sie sich nun an die Ergebnisse von PRINT CHR$(23);CHR$(1) usw. gewöhnen wollen, so stellen Sie sich unter dem linken Operanden bitte den Hintergrund, unter dem rechten Operanden das Grafikzeichen, und unter einer Eins einen gesetzten Punkt vor.
Was dann nach einer Print-Anweisung übrig bleibt, können Sie rechts des > - Zeichens ablesen.


AND
OR
XOR

Die Zusammenhänge deutlich machen auch folgende Zeilen:

```
10 MODE 0
20 LOCATE 1,1
30 INPUT "0,1,2 ODER 3 FUER *";E
40 PRINT CHR$(23);CHR$(E)
50 TAG
60 MOVE 100,100
70 PRINT "*";
80 TAGOFF
90 LOCATE 1,2
100 INPUT"0,1,2 ODER 3 FUER #";E1
110 PRINT CHR$(23);CHR$(E1)
120 TAG
130 MOVE 100,100
140 PRINT "#";
150 TAGOFF
160 GOTO 20
```

Wie Sie sehen dürfte für unsere Zwecke wohl die 1, die OR-Verknüpfung der Punkte in Frage kommen.

Denn mit der ersten Print-Anweisung werden alle Punkte gesetzt, die in beiden Zeichen vorkommen. Wird dann eines der Zeichen noch einmal an genau der gleichen Position ausgeben, dann muß das Resultat von OR das Andere sein.

So kann denn auch unsere Lokomotive nun an einem Bahnhof vorbeifahren, welcher hier allerdings nur symbolisch angedeutet wird.

```
10 REM zug4
20 REM "spritedemo"
30 REM -----------
40 MODE 2:DEFINT a-z
50 SYMBOL AFTER 129
60 SYMBOL  130, 1, 3, 7, 15, 31, 4, 3, 0
70 SYMBOL  131, 255, 255, 255, 255, 255,
 128, 0, 0
80 SYMBOL  132, 254, 254, 254, 255, 254,
 72, 48, 0
90 MODE 2
100 ORIGIN 1,1
110 PLOT 1,35: DRAW 640,35
120 MOVE 100,50:TAG:PRINT"HINTERGRUND";
130 zug$=CHR$(130)+CHR$(131)+CHR$(132)
140 TAGOFF
150 PRINT CHR$(23);CHR$(1)
160 TAG
170 FOR x=701 TO -29 STEP -10
180 MOVE x,50
190 CALL &BD19
200 PRINT zug$;
210 MOVE x+10,50
220 PRINT zug$;
230 NEXT x
240 GOTO 170
```

Anwendung: Arcade-Game

Das folgende Beispielprogramm zeigt noch einmal die Anwendung einiger der eben vorgestellten Techniken.
Es realisiert ein einfaches Schießspiel von der Art 'Duckshot' oder 'Scheibenschießen', die genaue Bezeichnug wird wohl von der Vorstellungsgabe des jeweiligen "Täufers" abhängen.

Ziel der eben genannten Spiele ist es, möglichst viele aus einer Reihe sich von links nach rechts über den Bildschirm bewegenden Zielen abzuschießen.
Zu diesen Zweck stehen dem Spieler mehrere Geschosse zur Verfügung, die er von einer beweglichen Basis aus abschießen kann.
Außerdem werden, um den Spieler zu motivieren, während des ganzen Spieles die erzielte Gesamtpunktzahl wie auch die Zahl der noch zur Verfügung stehenden Geschosse angezeigt.

Bei einem solchen Spiel handelt es sich gewissermaßen um ein Standardprogramm, denn es kann nach den jeweiligen Erfordernissen umgeschrieben werden.

Alle wichtigen Elemente der Spieleprogrammierung lassen sich finden:
So wird die Basis des Spielers in Abhängigkeit von den Tasten 'Z' und ' ' gesteuert, doch könnte es stattdessen auch ein Raumschiff sein, welches, ähnlich dem Cursor unseres Zeicheneditors, in mehr als nur zwei Richtungen gesteuert wird.
Denkbar ist es auch, daß, so wie hier die Raketen des Spielers aufsteigen, sich 'Space Invaders' vom oberen Bildschirmrand herabsenken.
Und auch die SCORE-Routine kann als Anregung für die Spielerbewertung in einem anderen Spiel genommen werden; hier beispielsweise sind die erzielten Punkte abhängig von der Geschwindigkeit, mit der das Ziel sich bewegt.

```
10 DEFINT r,s
20 DEFREAL g,z
30 MODE 1
40 s=20:zs=1:rz=1:schussmax=20
50 GOSUB 300
60 e$=INKEY$
70 sbalt=s
80 LOCATE zs,rz:PRINT" ";:REM ziel loesc
hen
90 IF ziel=0 THEN rz=RND(1)*20+0.5:gesch
=RND(1)*2:ziel=-1:zs=1:REM neues Ziel
100 zs=zs+gesch:IF zs>38 THEN ziel=0
110 CALL &BD19
120 LOCATE zs,rz:PRINT CHR$(225);:REM zi
el
130 IF e$="z" THEN s=s-1:IF s<1 THEN s=1
140 IF e$="\" THEN s=s+1:IF s>40 THEN s=
40
150 IF (e$=" " AND schuss=0) THEN schuss
=-1:ss=s:rs=22:schusszaehler=schusszaehl
er+1:GOSUB 300
160 IF schuss=-1 THEN GOSUB 220
180 LOCATE sbalt,23:PRINT" ";:REM basis
loeschen
190 LOCATE s,23:PRINT"^";:REM basis
200 GOTO 60
210 REM schussbehandlung
220 LOCATE ss,rs:PRINT" ";:REM schuss lo
eschen
230 IF (rs=rz AND ss=INT(zs)) THEN punkt
=INT(punkt+100*gesch):schuss=0:ziel=0:GO
SUB 300:PRINT CHR$(7):GOTO 260
240 rs=rs-1:IF rs=0 THEN schuss=0:GOTO 2
50 ELSE LOCATE ss,rs:PRINT CHR$(239);:RE
M schuss
250 IF (schuss=0 AND schusszaehler=schus
smax) THEN GOTO 270
```

```
260 RETURN
270 LOCATE 9,10:PRINT"* * * GAME OVER *
* *";
280 e$=INKEY$:IF e$="" THEN 280
290 IF e$=CHR$(13) THEN RUN ELSE 280
300 LOCATE 1,24:PRINT"SCORE: ";punkt:LOC
ATE 29,24:PRINT"SCHUSS: ";schussmax-schu
sszaehler;
310 RETURN
```

In Zeile 40 werden zunächst die Variablenwerte zu Spielbeginn festgelegt. Dabei bedeuten

s = Bildschirmspalte der Basis
zs = Bildschirmspalte des Zieles
rz = Bildschirmreihe des Zieles

Anschließend wird ein Unterprogramm ab Zeile 300 aufgerufen, welches den Spielstand in der untersten Bildschirmzeile anzeigt.
Nachdem dann eine eventuelle Aktion des Spielers in Zeile 60 festgestellt wurde, wird die alte Position der Spielerbasis notiert, damit ein korrekter Löschvorgang ihres alten Abbildes später durchgeführt werden kann.

Zeile 90 wertet dann ein Flag aus, welches anzeigt, ob sich noch ein Ziel auf dem Bildschirm befindet. Sollte dies nicht der Fall sein, werden Reihe wie auch Geschwindigkeit eines Geschosses willkürlich festgelegt.
Nach der Berechnung der neuen Bildschirmkoordinate und einem Test auf eine eventuelle Bereichsüberschreitung, wird anschließend das Ziel auf dem Bildschirm positioniert.
Dann erfolgt die Auswertung der zuvor übernommenen Tastaturwerte - wollte der Spieler seine Basis nach rechts oder links bewegen, oder wollte er mittels der Leertaste einen Schuß abgeben ?

Im letzten Fall werden die verschiedenen Zähler auf den aktuellen Stand gebracht, und ein weiteres Flag wird gesetzt, so daß das Unterprogramm zur Bewegung der abgeschossenen Rakete (220) nicht in jedem Fall angesprochen werden muß.

Die letzten beiden Arbeiten des Hauptprogrammes bestehen dann darin, die Spielerbasis zu löschen und an der neuen Position wieder auszugeben.

Anwendung: Grafik Editor

Bevor Ihnen im weiteren Verlauf dieses Buches nun Mathematik und Technik der zwei- und dreidimensionalen Grafikprogrammierung nahegebracht werden, soll Ihnen zunächst noch ein weiteres Programm vorgestellt werden welches ebenso wie der Zeichensatz-Programmierer zu einem unentbehrlichem Hilfsmittel werden kann.

Es ist die Rede von einem Grafik-Editor, von einem Programm, das es Ihnen erlaubt, mit dem Cursor und dem Monitor umzugehen wie mit Bleistift und Papier.
Doch wollen wir es nicht bei der einfachen Zeichenfunktion belassen, sondern der Editor soll sich auch als Programmierer betätigen und auf Knopfdruck vollständige Basic-Programme erstellen, die Sie dann, wie schon die Symbol-Tabellen, mit MERGE an Ihre Werke anbinden können.

Wie praktisch diese Anwendung sein kann, werden Sie spätestens dann erfahren, wenn Sie mit den eben besprochenen Techniken ein Spiel programmiert haben und dieses dann mit einem geeigneten Hintergrund versehen wollen.
Sie werden sehen, daß, wenn Sie versuchen, eine vielleicht auf Papier bestehende Zeichnung in ein Unterprogramm umzusetzen, die parallelen Linien immer mehr oder weniger windschief sind, und zwar nur aus dem Grunde, weil es Ihnen nicht gelingt, aus 128 0000 möglichen Punkten den einzig richtigen anzusprechen.

Die wesentliche Aufgabe des Programmes wird es also sein, über sämtliche Koordinaten genau Buch zu führen.
So richten wir zu diesem Zweck mehrere Tabellen ein, welche für die Speicherung der Inhalte der Variablen x und y eines jeden gesetzten Punktes zuständig sind (60).
Außerdem soll der Editor sich merken, ob ein Punkt eben nur ein Punkt oder vielleicht das Ende einer Geraden ist,

weshalb wir die Variable modus$ ergänzen. Xm und ym sind Hilfsvariablen und werden nur in den wenigen Fällen belegt, wenn Sie einen Kreis gezeichnet haben.

Dem Hauptprogramm überlassen wir dann die Arbeit, Ihre Eingabe anzunehmen und auszuwerten.
Meistens werden Sie nur den Cursor bewegen wollen, so daß die Verzweigung zu diesem Unterprogramm zuerst durchgeführt wird.

Das Setzen eines Punktes wie auch das Zeichnen einer Linie erfordert keinen großen Programmieraufwand und kann deshalb auch sogleich nach der Identifikation dieses Wunsches durchgeführt werden (530,540).
Die übrigen Routinen, wie das Zeichnen von Kreisen, das Programmieren des Bildes oder auch das Abspeichern und Laden der Bilddaten, wurden als Unterprogramme ausgeführt.

Deren Aufbau dürfte eigentlich jedem verständlich sein, zumal die benutzten Variablen sinngemäß benannt und die Bezeichnungen nicht abgekürzt wurden.

Von besonderem Interesse wäre vielleicht noch die Zeile 6030.
Hier wird innerhalb der Kreiszeichenroutine der Cursor bewegt (GOSUB 7010), anschließend wird aus dem x- und y-Versatz zum Kreismittelpunkt der Radius berechnet, was wie auf Seite 37 erläutert geschieht.

```
10 REM **********************************
20 REM * Grafik Programmer; Version 1.0*
30 REM **********************************

40 MODE 1:PAPER 0:PEN 1:INK 0,12:BORDER
12:INK 1,0
50 DEFINT a-z:ORIGIN 1,1
60 DIM x(200),y(200),modus$(200),xm(200)
,ym(200)
70 c=10:farbe=1
80 REM PEN 1:INK 1,12
90 LOCATE 7,5:PRINT"Grafik Editor & Prog
rammierer"
100 LOCATE 13,6:PRINT"by J";CHR$(178);"r
g Walkowiak"
110 LOCATE 1,12:PRINT STRING$(40,154)
120 LOCATE 10,20:PRINT CHR$(164);" 1985
by DATA BECKER"
130 FOR i=1 TO 5000:NEXT
140 GOTO 1010
500 REM **************** Hauptprogramm
510 eingabe$=INKEY$
520 GOSUB 7010:REM cusor bewegen
530 IF eingabe$="p" THEN x(p)=x:y(p)=y:m
odus$(p)="p":PLOT x,y,farbe:p=p+1:lx=x:l
y=y:GOTO 510:REM *** punkt setzen
540 IF eingabe$="l" THEN  modus$(p)="l":
x(p)=x:y(p)=y:PLOT lx,ly,farbe:DRAW x,y,
farbe:lx=x:ly=y:p=p+1:GOTO 510:REM ***li
nie zeichnen
550 IF eingabe$="P" THEN GOTO 2010
560 IF eingabe$="z" THEN GOTO 5010
570 IF eingabe$="S" THEN 3010
580 IF eingabe$="L" THEN 4010
590 IF (eingabe$="h" OR eingabe$="H") TH
EN GOTO 1010
600 IF eingabe$="c" THEN CLS:INPUT"neue
```

```
Cursorschrittweite";c:GOTO 5010
610 IF eingabe$="@"THEN p=p-1:GOTO 5010
620 IF eingabe$="k" THEN 6000:REM zeichn
e kreis
630 GOSUB 8010:REM cursor blink
640 GOTO 510
650 REM ************* ende hauptprogramm
1000 REM ************************* inst
1010 MODE 1:CLS:LOCATE 15,1:PRINT"  Hilf
smenue":PRINT STRING$(40,154);
1020 PRINT"7  8  9            ";:PEN 2:PRIN
T"Cursorbewegung":PEN 1
1030 PRINT"4     6            entsprechend
der
1040 PRINT"1  2  3            Anordnung der
 Tasten                       auf dem 12-er
 Block"
1050 PRINT"   5               Cursor auf Sc
hirmmitte":PRINT
1060 PEN 2: PRINT"Zeichenbefehle";:PEN 1
:PRINT" mit Kleinbuchstaben:"
1070 PRINT" p - Punkt an Cursorposition
setzen"
1080 PRINT" l - Linie vom zuletzt gesetz
ten Punkt        aus zeichnen"
1090 PRINT" k - Kreismittelpunkt, ENTER
wenn             Radius gewaehlt
1100 PRINT" z - Bild neu zeichnen"
1110 PRINT" c - Cursorschrittweite setze
n"
1120 PRINT
1130 PEN 2:PRINT"Steuerbefehle";:PEN 1:P
RINT" mit Grossbuchstaben:"
1140 PRINT" H - Hilfsmenue
1150 PRINT" S - Bilddaten speichern
1160 PRINT" L - Bilddaten laden
1170 PRINT" P - Basicprogramm erzeugen"
```

```
1180 PEN 2:PRINT" @ - macht letzten Befe
hl unwirksam":PEN 1
1190 PRINT STRING$(40,154);:PRINT"Wenn O
.K. dann beliebige Taste druecken.";

1200 IF INKEY$="" THEN 1200
1210 CLS:x=320:y=200:GOTO 510
2000 REM ************************ prog
2010 CLS:PRINT"Grafik programmieren ..."
:PRINT STRING$(40,154)
2020 INPUT"Wie soll das Programm heissen
";name$:IF LEN(name$)>16 THEN GOTO 2010

2030 IF name$="@"THEN GOTO 5010
2040 PRINT:INPUT"Nummer der ersten Progr
ammzeile";zeile
2050 PRINT:INPUT"Welchen Zeilenabstand w
uenschen Sie";zeilenabstand:PRINT
2060 WINDOW #1,1,40,18,25
2070 OPENOUT name$
2080 zeile$="ORIGIN 1:1:DEG:GOTO "+STR$(
zeile+2*zeilenabstand):PRINT#1,zeile;zei
le$:PRINT#9,zeile;zeile$:zeile=zeile+zei
lenabstand
2090 zeile$="FOR grad=1 TO 360:PLOT r*CO
S(grad),r*SIN(grad):NEXT grad:ORIGIN 1,1
:RETURN":upkreis=zeile:PRINT#1,zeile;zei
le$:PRINT#9,zeile;zeile$:zeile=zeile+zei
lenabstand
2100 zeile$=""
2110 FOR i=0 TO p-1
2120 IF modus$(i)="p"THEN befehl$="plot"
:GOTO 2150
2130 IF modus$(i)="l"THEN befehl$="draw"
:GOTO 2150
2140 IF modus$(i)="k" THEN zeile$="r="+S
TR$(x(i))+":ORIGIN "+STR$(xm(i))+","+STR
```

```
$(ym(i))+":GOSUB "+STR$(upkreis):GOTO 21
60
2150 zeile$=befehl$+STR$(x(i))+","+STR$(
y(i))
2160 IF LEN(progzeile$)<2 THEN progzeile
$=zeile$:zeile$="":GOTO 2200
2170 IF (LEN(progzeile$)<200 AND LEN(pro
gzeile$)>5) THEN progzeile$=progzeile$+"
:"+zeile$:zeile$="":GOTO 2200
2180 PEN 3:PRINT#9,zeile;progzeile$:PEN
2:PRINT#1,zeile;progzeile$
2190 zeile=zeile+zeilenabstand:progzeile
$=zeile$
2200 NEXT i
2210 IF LEN(progzeile$)>1 THEN PEN 3:PRI
NT#9,zeile;progzeile$:PEN 2:PRINT#1,zeil
e;progzeile$
2220 PEN 3:CLOSEOUT
2230 PEN 1
2240 PRINT:INPUT"Programmlauf beenden ";
eingabe$:IF UPPER$(LEFT$(eingabe$,1))="J
" THEN MODE 1:END
2250 GOTO 5010
3000 REM *************************** save
3010 CLS:PRINT"Bilddaten speichern ...":
PRINT STRING$(40,"_")
3020 INPUT"Wie soll das Bild heissen";na
me$:IF LEN(name$)>16 THEN GOTO 3010
3030 IF name$="@"THEN GOTO 5010
3040 OPENOUT name$
3050 PEN 3:PRINT
3060 PRINT#9,p
3070 FOR i=0 TO p
3080 PRINT #9,x(i),y(i),modus$(i),xm(i),
ym(i)
3090 NEXT i
3100 CLOSEOUT
```

```
3110 PEN 1
3120 PRINT:INPUT"Programmlauf beenden ";
eingabe$:IF UPPER$(LEFT$(eingabe$,1))="J
" THEN MODE 1:END
3130 GOTO 5010
4000 REM *************************** load
4010 CLS:PRINT"Bilddaten laden ...":PRIN
T STRING$(40,"_")
4020 INPUT"Welches Bild";name$:IF LEN(na
me$)>16 THEN GOTO 4010
4030 IF name$="@"THEN GOTO 5010
4040 PRINT:PEN 2
4050 OPENIN name$
4060 INPUT#9,p
4070 FOR i=0 TO p-1
4080 INPUT#9,x(i),y(i),modus$(i),xm(i),y
m(i)
4090 NEXT i
4100 CLOSEIN
4110 PEN 1
5000 REM *************************** zeich
5010 ORIGIN 1,1:CLS:FOR i=0 TO p-1
5020 IF modus$(i)="p" THEN PLOT x(i),y(i
),farbe:GOTO 5040
5030 IF modus$(i)="l" THEN DRAW x(i),y(i
),farbe
5040 IF modus$(i)="k" THEN ORIGIN xm(i),
ym(i):DEG:radius=x(i):FOR grad=1 TO 360:
PLOT radius*COS(grad),radius*SIN(grad),f
arbe:NEXT grad:ORIGIN 1,1
5050 NEXT i
5060 GOTO 510
6000 REM *************************** kreis
6010 xm=x:ym=y:ORIGIN xm,ym:x=0:y=0:DEG

6020 eingabe$=INKEY$:GOSUB 8010:IF einga
be$="" THEN 6020
```

```
6030 IF eingabe$<>CHR$(13) THEN GOSUB 70
10:radius=SQR(x*x+y*y):GOSUB 8010:GOTO 6
020
6040 FOR grad=1 TO 360
6050 x=radius*COS(grad):y=radius*SIN(gra
d):PLOT x,y,farbe
6060 NEXT grad
6070 modus$(p)="k":x(p)=radius:y(p)=radi
us:xm(p)=xm:ym(p)=ym:p=p+1
6080 ORIGIN 1,1:x=xm:y=ym:GOTO 510
7000 REM ******************* up cursor
7010 IF eingabe$="8" THEN Y=Y+C:GOTO 710
0
7020 IF eingabe$="6"THEN X=X+C:GOTO 7100
7030 IF eingabe$="4"THEN X=X-C:GOTO 7100
7040 IF eingabe$="2"THEN Y=Y-C:GOTO 7100
7050 IF eingabe$="9"THEN x=x+c:y=y+c:GOT
O 7100
7060 IF eingabe$="1"THEN x=x-c:y=y-c:GOT
O 7100
7070 IF eingabe$="3"THEN x=x+c:y=y-c:GOT
O 7100
7080 IF eingabe$="5"THEN x=320:y=200:GOT
O 7100
7090 IF eingabe$="7"THEN x=x-c:y=y+c:GOT
O 7100
7100 RETURN
8000 REM ******************** up blink
8010 farbe1=TEST(x,y) :REM grafikcursor
blink
8020 FOR x1=1 TO 10
8030 PLOT x,y,farbe1+1
8040 PLOT x,y,farbe1
8050 NEXT x1
8060 RETURN
```

Bedienungshinweise zum Grafikeditor:

Die Steuerung sämtlicher Funktionen des Grafik-Editors erfolgt durch das Niederdrücken entsprechend definierter Tasten.
Es ist nicht erforderlich, eine Anweisung vor der Ausführung mittels ENTER zu bestätigen !

Generell lassen sich die Anweisungen in drei Gruppen einteilen:

1. Cursor-Befehle
Die Steuerung des Cursors erfolgt über die Zifferntasten der links neben dem Rekorder angebrachten 10-er Tastatur.
Dabei entspricht die Bewegungsrichtung des blinkenden Punktes der geometrischen Anordnung der Taste, d.h. 6 bewegt den Cursor nach rechts, 7 nach links oben, usw.

Eine Sonderfunktion kommt dabei der Ziffer 5 zu:
Sie positioniert den Cursor auf der absoluten Position 320,200, somit in der Bildmitte.

2. Steuerbefehle
Steuerbefehle werden immer als Großbuchstaben eingegeben, Sie dürfen also nicht vergessen, bei der Eingabe des Befehles auch die SHIFT-Taste niederzuhalten.

Unter die Steuerbefehle fallen das Speichern und Laden der Bilddaten, das Erzeugen eines Basic-Programmes oder auch das Neuzeichnen eines Bildes zwecks Korrektur.

3. Zeichenbefehle
werden sofort nach Eingabe des entsprechenden Kleinbuchstabens ausgeführt.

Darüber hinaus steht Ihnen immer die Hilfsfunktion zur Verfügung; nach Eingabe von H oder h gibt Ihnen das Programm auf einer Tafel noch einmal Auskunft über alle Möglichkeiten.

PUNKTE

Um einen Punkt zu zeichnen, steuern Sie den Cursor auf die gewünschte Position, und drücken **p** für Punkt.
Sofort wird der Punkt gesetzt und die entsprechenden Koordinaten werden für die spätere Programmierung gespeichert.

LINIEN

Steuern Sie den Cursor zunächst auf den Startpunkt der Linie und drücken Sie **p**. Anschließend bewegen Sie ihn an den gewünschten Endpunkt und drücken **l**.

Es wird immer eine Linie ausgehend von dem zuletzt gesetzten Punkt gezeichnet, dabei ist es gleichgültig, ob dieser mit p gesetzt wurde, oder ob es sich um den Endpunkt einer Linie handelte.

KREISE

Bewegen Sie den Cursor zunächst auf den gewünschten Mittelpunkt und drücken Sie die **k**-Taste.
Anschließend legen Sie mit den Cursortasten den Radius fest, wobei die Bewegungsrichtung keine Rolle spielt.
Ein weiterer Druck auf **k** führt den Zeichenvorgang aus.

FEHLERBEHANDLUNG

Sollte Ihnen während des Zeichnens ein Fehler unterlaufen sein, so müssen Sie dennoch nicht von vorne beginnen.
Nach Eingabe des AT-Zeichens, das ist die Taste rechts neben dem P, wird der Bildschirm gelöscht, und das Bild wird neu gezeichnet, allerdings wird dabei der zuletzt eingegebene Befehl nicht ausgeführt.
Diesen Vorgang können Sie so oft wiederholen, bis auch der zuerst gesetzte Punkt wieder vom Schirm verschwunden ist.

BILD SPEICHERN, BILD LADEN

Betätigen Sie **L** oder **S** und geben Sie einen bis zu acht Buchstaben langen Namen für das Bild ein.
Nachdem Sie dann die ENTER-Taste betätigt haben, wird der gewünschte Vorgang ausgeführt.

PROGRAMM GENERIEREN

Nach Niederhalten von **P** ist wiederum die Eingabe eines bis zu acht Buchstaben langen Namens erforderlich.
Außerdem werden Sie nach der Zeilennummer gefragt, mit der das zu erstellende Programm beginnen soll, wie auch nach der Schrittweite der einzelnen Zeilen.
Anschließend wird auf den Datenträger ein eigenständiges Programm geschrieben, welches Sie mit MERGE", LOAD" oder RUN" laden können.

Beachten Sie jedoch, daß der Editor dabei nicht die Betriebsart festlegt, da die so erzeugten Programme als Subroutinen gedacht sind.
Aus diesem Grunde steht zu Beginn des erzeugten Programmes auch keine CLS-Anweisung, die Sie ergänzen müßten, wenn es Ihnen allein um den Aufbau der Grafik geht.

4. KAPITEL
2D - GRAFIK

Techniken

Bislang wurden nur die grundsätzlichen Grafikbefehle des Schneider CPC sowie Techniken zur Bildschirmmanipulation vorgestellt.
So wurde gezeigt, wie die gewünschte Farbe gewählt, wie Grafikelemente auf den Schirm bebracht und überlagert oder einfache Formen gezeichnet werden können.
Zwar leutet es ein, daß alle gewünschten Grafiken sich allein durch das Setzen von Punkten realisieren lassen, doch wäre die Erstellung von Bildern allein mit diesen Anweisungen ein ebenso mühseliges wie auch zeitaufwendiges Unterfangen.
Und der Versuch einer Programmierung von Anwendungsprogrammen, welche die so erstellten Grafiken manipulieren, wäre von vornherein zum Scheitern verurteilt.

Man muß sich daher nach geeigneten Hilfsmitteln umsehen, und findet diese dann auch im Bereich der Mathematik.
Hier lassen sich Formeln finden, welche gewissermaßen als Patentrezepte angesehen werden können, und in die nur die entsprechenden Zahlenwerte eingesetzt werden müssen.

Im Rahmen dieses Kapitels sollen daher einige mathematische Grundlagen vorgestellt und durch Beispielprogramme belegt werden.

Shapes

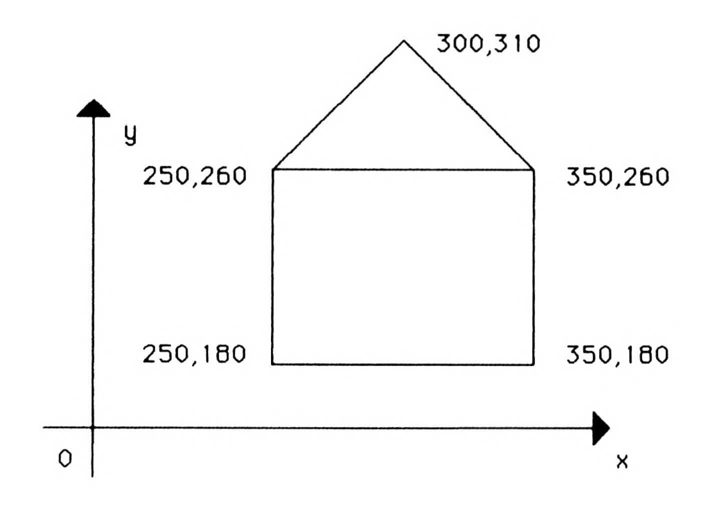
lassen sich in den verschiedensten Heim- und Hobbycomputern finden.
So auch bei unserem Schneider CPC 464.
Zwar kennt das Locomotive Basic keinen entsprechenden Befehl, und der Leser des beigefügten Handbuches wird auch nirgends auf dieses Wort stoßen, doch läßt sich die Technik

der Shapes dank der relativen Plotbefehle einfach anwenden.

Im Sinne des Erfinders ist ein Shape die Beschreibung einer Figur durch deren Eckpunkte, die dann durch Linien miteinander verbunden werden.
Der äußere Umriß des Hauses auf der folgenden Seite kann also mit

250,180
250,260
300,310
350,260
350,180
250,180

beschrieben werden.



Um das Bild des Hauses auf den Schirm zu bringen, werden die meisten Computerbesitzer wohl das folgende Programm benutzen:

```
10 MODE 1:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
20 PLOT 250,180:   REM UNTEN LINKS
30 DRAW 250,260:   REM OBEN LINKS
40 DRAW 300,310:   REM DACHSPITZE
50 DRAW 350,260:   REM OBEN RECHTS
60 DRAW 350,180:   REM UNTEN RECHTS
70 DRAW 250,180:   REM UNTERKANTE
```

Eine alternative Methode bewahrt die Bilddaten innerhalb eines Blockes auf, anstatt sie über mehrere Anweisungszeilen zu verteilen.
Die einzelnen Linien werden dann auch nicht mehr direkt, sondern von einem speziellen Programmteil gezeichnet:

```
10 MODE 1:PAPER 0:INK 0,12:BORDER  12:PEN
 1:INK 1,0
20 READ X,Y: REM ERSTER PUNKT
30 PLOT X,Y
40 READ X,Y: REM WEITEREN PUNKT LESEN
50 DRAW X,Y: REM LINIE ZEICHNEN
60 GOTO 40:  REM UND WEITER

1000 REM NACHFOLGEND DIE BILDDATEN
1020 DATA 250,180,250,260,300,310,350,260
1030 DATA 350,180,250,180
```

Noch bricht das Programm mit einer Fehlermeldung ab, da ihm die Anzahl der Eckpunkte nicht bekannt war, doch läßt sich dieser Umstand, zumal es sich bei Shapes fast immer um vordefinierte Figuren handelt, schnell beheben.

```
10 MODE 1:PAPER 0:INK 0,12:BORDER  12:PEN
 1:INK 1,0
20 READ ANZ: REM ANZAHL PUNKTE
30 READ X,Y: REM ERSTER PUNKT
40 PLOT X,Y: REM UND SETZEN
50 FOR PUNKT=1 TO ANZ-1
60 READ X,Y
70 DRAW X,Y
80 NEXT PUNKT
90 END

1000 REM NACHFOLGEND DIE BILDDATEN
1010 DATA 6
1020 DATA 250,180,250,260,300,310,350,260
1030 DATA 350,180,250,180
```

Das erste Element des Datenblockes gibt an, aus wie vielen Elementen das Shape existiert, denn um nichts anderes als um die Shapedaten handelt es sich bei den Programmzeilen von 1010 bis 1030.

Vielleicht erscheint Ihnen diese Konstruktionsweise für das einfache Zeichen irgendwelcher Figuren zu aufwendig, doch beachten Sie, wie leicht sich die Bilder austauschen lassen:

```
10 MODE 1:PAPER 0:INK 0,12:BORDER  12:PEN
 1:INK 1,0
20 READ ANZ: REM ANZAHL PUNKTE
30 READ X,Y: REM ERSTER PUNKT
40 PLOT X,Y: REM UND SETZEN
50 FOR PUNKT=1 TO ANZ-1
60 READ X,Y
70 DRAW X,Y
```

```
80 NEXT PUNKT
90 END
1000 REM SHAPEDATEN
1010 DATA 11,320,240,330,230,330,180,340
1020 DATA 170,330,170,320,180,310,170,30
0
1030 DATA 170,310,180,310,230,320,240
```

Ebenso ist das Programm, nach einer kleinen Änderung, in der Lage, beliebig viele Shapes, und somit komplette Bilder, zu erzeugen.
Wir müssen nur ein Kennzeichen vereinbaren, welches das Datenende vereinbart.

In unserem Falle, solange der Koordinatenursprung noch bei 0,0 liegt, kann eine negative Zahl gewählt werden. Sind von der Konstruktion des Koordinatensystemes diese als gültige Angaben erlaubt, ist ein Wert zu wählen, der außerhalb der Zeichenfläche liegt.

```
10 MODE 1:PAPER 0:INK 0,12:BORDER  12:PEN
 1:INK 1,0
20 READ ANZ: REM ANZAHL PUNKTE
30 READ X,Y: REM ERSTER PUNKT
35 IF X<0 THEN END
40 PLOT X,Y: REM UND SETZEN
50 FOR PUNKT=1 TO ANZ-1
60 READ X,Y
70 DRAW X,Y
80 NEXT PUNKT
90 GOTO 20

1000 REM NACHFOLGEND DIE BILDDATEN
1010 DATA 6
1020 DATA 250,180,250,260,300,310,350,260
```

```
1030 DATA 350,180,250,180
1040 REM DACH
1050 DATA 2,250,260,350,260
1060 REM FENSTER
1070 DATA 5,340,250,320,250,320,230,340
1080 DATA 230,340,250
1090 DATA 5,260,250,260,230,280,230,280
1100 DATA 250,260,250
1110 REM TUER
1120 DATA 4,290,180,290,210,310,210,310
1130 DATA 180,-1
```

Koordinatentransformation

Nun sollten Sie voranstehendes Programm folgendermaßen ändern:

```
15 FOR FAKTOR=0.1 TO 1.9 STEP 0.2
16 RESTORE
25 IF ANZ<=0 THEN NEXT FAKTOR
40 PLOT X*FAKTOR,Y*FAKTOR
70 DRAW X*FAKTOR,Y*FAKTOR
```

Wie Sie sehen erhalten Sie gleich eine Reihe von Häusern, wobei eins größer als das andere ist.
Die Skalierung, das Vergrößern und Verkleinern von Objekten, läßt sich also auf einfache Weise durch Multiplikation mit einem konstanten Faktor bewerkstelligen. Mathematisch betrachtet handelt es sich bei diesem Vorgang um eine Koordinatentransformation, die anhand von Matrizen dargestellt werden kann.
Dadurch eröffnen sich dem Anwender noch eine Reihe von weiteren Möglichkeiten, die wir nun im einzelnen untersuchen wollen.

Das Koordinatensystem

Verschiebungen, Vergrößerungen, Verkleinerungen und auch Spiegelungen lassen sich im zweidimensionalem Fall ohne allzu aufwendige Berechnungen im kartesischen Koordinatensystem bewerkstelligen.
Um jedoch die Auswirkungen der verschiedensten Manipulationen auf dem Bildschirm beobachten zu können, ist es nicht mehr ausreichend, nur innerhalb des ersten Quadranten zu arbeiten.

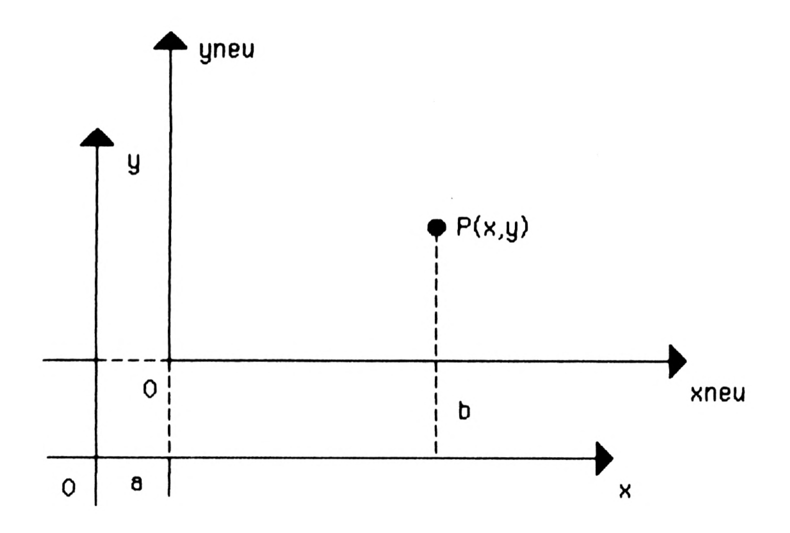
Aus diesem Grunde legen wir für alle folgenden Programme den Koordiantenursprung in der Mitte des Bildschirmes fest und zeichnen als Orientierungshilfe auch die beiden Achsen ein.

```
10 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
20 ORIGIN 320,200
100 TAG
110 PLOT -320,0:DRAW 320,0
120 MOVER -5,6:PRINT CHR$(246);
130 MOVER -17,-10:PRINT "x";
140 PLOT 0,-200:DRAW 0,200
150 MOVER -4,-2:PRINT CHR$(244);
160 MOVER -17,-10:PRINT "y";
170 TAGOFF
```

Gezeichnet werden die Achsen in den Zeilen 110 und 140. Bei der Abarbeitung der Zeilen 120 und 150 wird zunächst der Grafikcursor von der Stelle des zuletzt gesetzten Punktes ein wenig zurückgesetzt, anschließend werden die Strahlspitzen ausgegeben. Gleiches geschieht in den Zeilen 139 und 160, die der Beschriftung der Achsen dienen.
Diese Besonderheit, nämlich das Schreiben mittels der normalen Printanweisung an sämtlichen Grafikkoordinaten, wird mittels der Anweisung TAG aufgerufen. Mit TAGOFF kann dann jederzeit wieder auf normale Betriebsart zurückgeschaltet werden.

Verschiebungen

Die Parallelverschiebung ist die einfachste Art der Transformation eines kartesischen Koordinatensystemes.
Sie besteht aus einer Verschiebung der Achsen ohne deren Richtungen zu ändern.



Obiger Skizze läßt sich leicht entnehmen, daß die Größe der Verschiebung durch die Summanden a und b festgelegt wird.
Der Zusammenhang zwischen dem alten und dem neuen Koordinatensystem läßt sich also wie folgt formulieren:

xneu = x - a

yneu = y - b

Da wir aber nicht die Achsen sondern den Punkt verschieben wollen, lauten die für uns richtigen Gleichungen

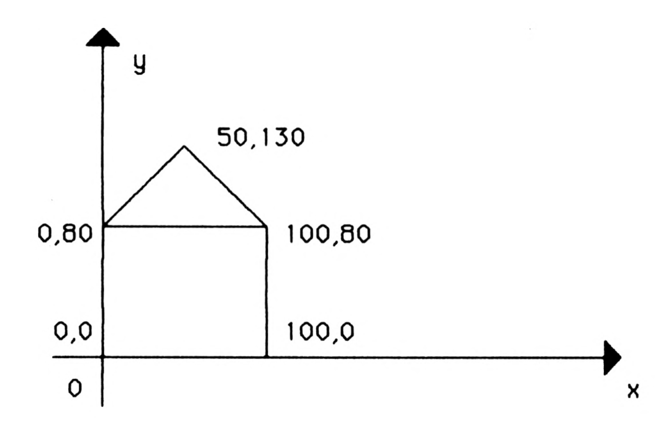
xneu = x + a

yneu = y + b

Mit diesen beiden Transformationsformeln können wir dann jeden Punkt in der Ebene verschieben; wollen wir mit räumlichen Bildnissen arbeiten, nehmen wir entsprechende Berechnungen für die dritte Achse vor:

zneu = z + c (zneu = z - c)

Daß alles seine Richtigkeit hat, beweisen wir wieder mit dem Umriß unseres Hauses. Allerdings müssen wir dabei zunächst beachten, daß die Koordinaten sich durch die Beziehung auf das neue Koordinatensystemes verändert haben:

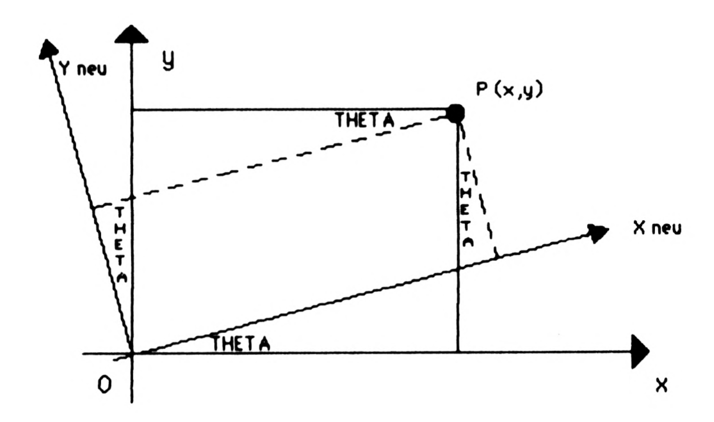


```
5 REM parallelverschiebung
10 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
20 ORIGIN 320,200
30 WINDOW#1,1,35,20,25
99 REM koordinatenachsen
100 CLG:TAG
110 PLOT -320,0:DRAW 320,0
120 MOVER -5,6:PRINT CHR$(246);
130 MOVER -17,-10:PRINT"x";
140 PLOT 0,-200:DRAW 0,200
150 MOVER -4,-2:PRINT CHR$(244);
160 MOVER -17,-10:PRINT"y";
170 TAGOFF
180 IF w=5 THEN 500
199 REM bildpunkte einlesen
200 RESTORE:READ anz
210 FOR punkt=1 TO anz
220 READ x(punkt):READ y(punkt)
230 NEXT punkt
240 IF w=1 THEN GOSUB 900
500 CLS#1
510 PRINT#1,"1 - Originalbild
520 PRINT#1,"2 - verschieben x"
530 PRINT#1,"3 - verschieben y"
540 PRINT#1,"4 - verschieben x & y"
550 PRINT#1,"5 - Bild loeschen"
590 INPUT#1,"Sie wuenschen";w
600 ON w GOTO 100,700,750,700,100
690 REM verschieben in x-richtung
700 INPUT#1,"Faktor x ";a
710 FOR punkt=1 TO anz
720 x(punkt)=x(punkt)+a
730 NEXT punkt
735 IF w=4 THEN 750
740 GOSUB 900:GOTO 500
750 INPUT#1,"Faktor y ";b
```

```
760 FOR punkt=1 TO anz
770 y(punkt)=y(punkt)+b
780 NEXT punkt
790 GOSUB 900:GOTO 500
800 CLG:GOTO 500
899 REM bild zeichnen
900 PLOT x(1),y(1)
910 FOR punkt=1 TO anz
920 DRAW x(punkt),y(punkt)
930 NEXT
940 RETURN
999 REM bilddaten
1000 DATA 8,0,0,0,80,50,130,100,80,100,0
1010 DATA 0,0,0,80,100,80
```

Drehungen

lassen sich fast ebenso einfach durchführen. In der Ebene versteht man darunter die gleichzeitige Drehung der beiden Koordinatenachsen um den Ursprungspunkt 0,0, wobei der Drehwinkel THETA den Grad der Drehung festlegt.



Wenn Sie die Ausführungen zur Konstruktion von Kreisen gelesen haben, so werden Sie rasch die Beziehung zwischen den Punkten x und xneu, wie auch y und yneu feststellen:

```
xneu = x * cos(theta)   +   y * sin(theta)

yneu = -x * sin(theta)  +   y * cos(theta)
```

Matrizen

Bei der Drehung wie auch bei der Verschiebung handelt es sich um sogenannte lineare Transformationen.
Diese werden gerne in sogenannter Matrizenform, in Zahlentafeln, dargestellt.
Dabei erfolgt die Anordnung der Zahlen, die nun als Matrixelemente bezeichnet werden, in Reihen und Spalten. Jede Zahl der Matrix kann dabei über ihre Position innerhalb des Schemas angesprochen werden, wobei die Indizierung nach dem gleichen Schema wie bei den Arrays in Basic erfolgt - nur auf das Komma wird verzichtet.

$$M = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}$$

Die Matrix selbst ist ein reines Zahlenschema, man kann nur die Anordnung der einzelnen Elemente angeben, nicht aber einen Gesamtwert berechnen.
Dennoch, findige Mathematiker haben nicht eher geruht, bis ganze Regale in Universitätsbibliotheken mit Rechenregeln für Matrizen gefüllt waren.
Selbst nur die wichtigsten Regeln und Ausnahmefälle zu nennen, würde den Rahmen dieses Buches sprengen, schließlich wollen Sie keine mathematischen Abhandlung lesen, sondern mehr zur Grafikprogrammierung erfahren.
Für unsere speziellen Fälle werden jedoch die Addition und besonders die Multiplikation von rechteckigen Matrizen wichtig, so daß beide Operationen an einem Beispiel vorgeführt werden sollen.

Betrachten wir die beiden Matrizen M1 und M2:

$$M1=\begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \qquad M2=\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Bei der Addition (Subtraktion) addiert (subtrahiert) man nun die Elemente mit demselben Zeilen- wie auch Spaltenindex. In unserem Falle lautet das Ergebnis daher:

$$\begin{bmatrix} -1 & 1 \\ 2 & 1 \end{bmatrix}$$

Die Multiplikation mutet auf den ersten Blick etwas seltsam an, erweist sich nach einiger Übung aber auch nicht mehr als Problem:
Man multipliziert die Elemente einer Zeile der ersten Matrix mit den Elementen einer Spalte der zweiten Matrix. Die resultierenden Produkte werden anschließend addiert:

$$\begin{matrix} -1 * 0 + 0 * 1 & \qquad & -1 * 1 + 0 * 1 \\ 1 * -1 + 0 * 1 & \qquad & 1 * 1 + 0 * 1 \end{matrix}$$

$$\begin{matrix} 0 & -1 \\ -1 & 1 \end{matrix}$$

Allgemein ausgedrückt ergibt sich somit folgende Rechenvorschrift:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} A*a+B*c & A*b+B*d \\ C*a+D*c & C*b+D*d \end{bmatrix}$$

Eine Transformation wie die eben durchgeführte Parallelverschiebung läßt sich nun recht gut als Produkt zweier Matrizen darstellen, allerdings handelt es sich dann um eine Multiplikation zwischen einer 1 x 2 Matrix und einer 2 x 2 Matrix:

$$x \quad y \quad * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a*x+c*y \quad b*x+d*y$$

X und y sind dabei die Koordinaten eines jeden Originalpunktes, die Summen auf der rechten Seite entsprechen xneu und yneu.

Welcher Art die durchgeführte Transformation ist, wird durch die Werte der zweiten Matrix festgelegt; die Möglichkeiten zeigt das folgende Programm.

```
5 REM matrixdemo
10 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN
 1:INK 1,0
20 ORIGIN 320,200
30 WINDOW#1,1,35,20,25
99 REM koordinatenachsen
100 CLG:TAG
110 PLOT -320,0:DRAW 320,0
120 MOVER -5,6:PRINT CHR$(246);
130 MOVER -17,-10:PRINT"x";
140 PLOT 0,-200:DRAW 0,200
150 MOVER -4,-2:PRINT CHR$(244);
160 MOVER -17,-10:PRINT"y";
170 TAGOFF
199 REM bildpunkte einlesen
200 RESTORE:READ anz
210 FOR punkt=1 TO anz
220 READ x(punkt):READ y(punkt)
230 NEXT punkt
500 CLS#1
510 INPUT#1,"a ";a
520 INPUT#1,"b ";b
530 INPUT#1,"c ";c
540 INPUT#1,"d ";d
550 CLS#1
560 PRINT#1,"Matrix:"
570 PRINT#1, a,b
580 PRINT#1, c,d
590 INPUT#1,"Zeichnen ?";e$
600 IF LOWER$(LEFT$(e$,1))<>"j" THEN 500
610 CLG:TAG
620 FOR punkt=1 TO anz
630 xneu(punkt)=a*x(punkt)+c*y(punkt)
640 yneu(punkt)=b*x(punkt)+d*y(punkt)
650 NEXT punkt
660 PLOT xneu(punkt),yneu(punkt)
670 FOR punkt=1 TO anz
```

```
680 DRAW xneu(punkt),yneu(punkt)
690 NEXT punkt
700 PLOT -320,0:DRAW 320,0
710 MOVER -5,6:PRINT CHR$(246);
720 MOVER -17,-10:PRINT"x";
730 PLOT 0,-200:DRAW 0,200
740 MOVER -4,-2:PRINT CHR$(244);
750 MOVER -17,-10:PRINT"y";
760 TAGOFF:GOTO 500
770 REM bild zeichnen
900 PLOT x(1),y(1)
910 FOR punkt=1 TO anz
920 DRAW x(punkt),y(punkt)
930 NEXT
940 RETURN
1000 DATA 8,0,0,0,80,50,130,100,80,100,0
1010 DATA 0,0,0,80,100,80
```

Nachdem Sie das Programm Matrixdemo korrekt eingegeben und gestartet haben, werden die Daten unseres Hausumrisses zunächst in den indizierten Variablen x und y abgespeichert (Zeile 200 bis 230).
Um die Eingaben des Benutzers entgegennehmen zu können, wurde innerhalb des dritten Quadranten ein Fenster definiert, in welchem Sie das Programm anschließend zur Eingabe von a, b, c und d auffordert.
Im folgenden wird das Schema der Matrix ausgedruckt, und sofern Sie sich nicht vertippt haben und mit der Matrix einverstanden sind, können Sie nach Eingabe von j und Drücken auf ENTER das veränderte Bildnis des Hauses zeichnen lassen.

Probieren Sie zunächst die verschiedensten Eingaben aus und versuchen Sie, Gesetzmäßigkeiten festzustellen.
Allerdings ist es dabei zweckmäßig, den Wertebereich auf Zahlen zwischen -2 und 2 zu begrenzen, außerdem sollten Sie systematisch vorgehen und nicht gleich alle vier Parameter mit Zufallsszahlen belegen.

Maßstabsänderungen, Skalierungen

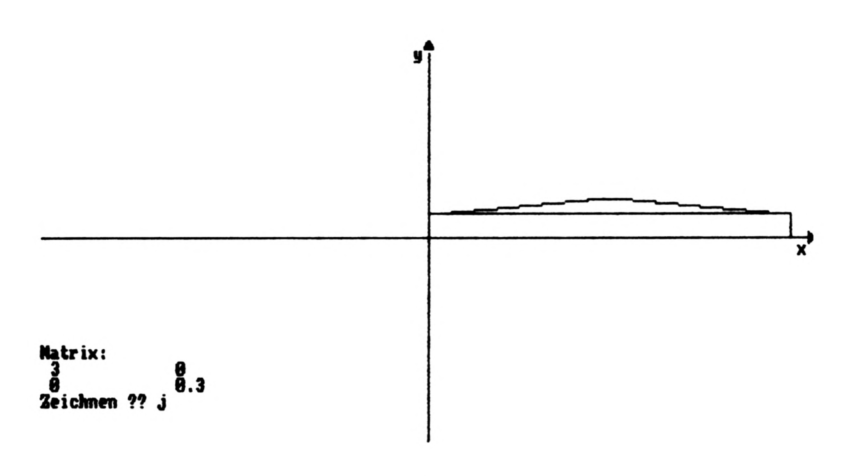
Wir hatten bereits festgestellt, daß die Multiplikation sämtlicher x-Werte mit einem konstanten Faktor das Bild in Richtung der x-Achse vergrößert (gleiches gilt natürlich auch für alle y(i) und die y-Achse).
Entsprechend muß die Multiplikation mit einer Zahl kleiner eins zu einer Verkleinerung des Bildes führen.

Betrachten wir nun unsere Matrix unter diesem Aspekt, so stellen wir fest, daß diese Faktoren wohl mit a und d identisch sind (setzen Sie für b und c Null ein):

$$x \quad y \quad * \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = a*x+0*y,0*x+d*y = a*x,b*y$$

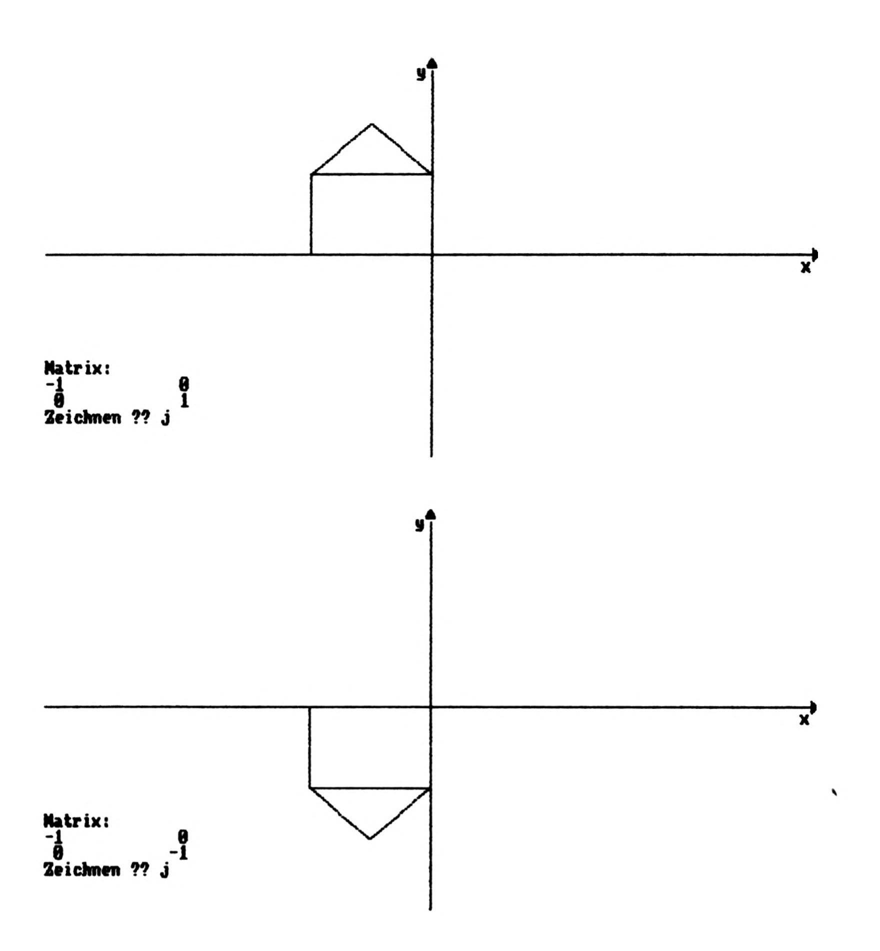
Mit den Werten für a und d legen wir also die Größe des Bildnisses fest, wobei a für eine Vergrößerung/Verkleinerung in x-Richtung, und d für eine Maßstabsänderung entlang der y-Achse stehen.
Dabei bedeuten Werte zwischen 0 ... 1 eine Verkleinerung und Werte zwischen 1 ... unendlich eine Vergrößerung.

Die folgende Abbildung zeigt somit ein Haus von dreifacher Breite und nur einem drittel der Originalhöhe.



So läßt sich auch eine 'neutrale Matrix' festlegen, denn wenn beide Faktoren gleich eins sind, wird die Abbildung identisch mit dem Originalbild sein.

Das Ergebnis dieser Matrizenmultiplikation wird dann eine **Spiegelung** an der y-Achse, der x-Achse, durch den Koordinatenursprung sein.

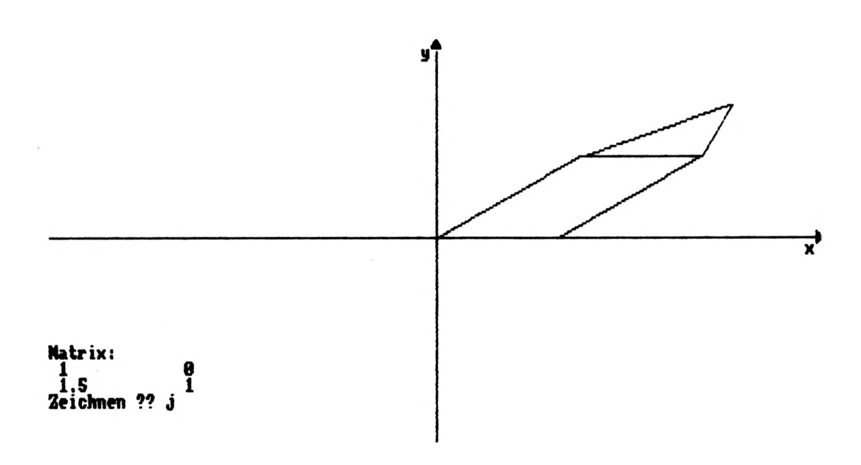


Interessant wird es nun, wenn Sie sich daran machen, die verschiedensten Werte für b und c einzusetzen. Hierbei entsteht fast schon ein räumlicher Eindruck, erscheinen die auf den Bildschirm projizierten Abbildungen doch wie ein Schattenriß des Originalbildes.

Auch hier wollen wir wieder das Schema zur Berechnung der einzelnen Koordinaten angeben:

$$x \quad y \quad * \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} = x, b*x+y$$

Da a und d beide gleich eins sind, wird keine Maßstabsänderung vorgenommen, wohl aber wird durch das Matrixelement b die y-Achse mit wachsendem x zunehmend verschoben.
Entsprechendes gilt für die x-Achse, wenn c ungleich Null gewählt wird.

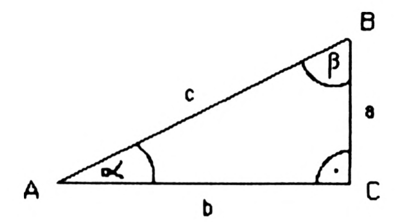


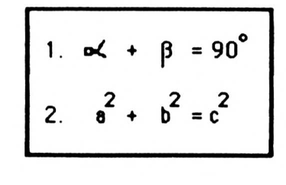
Rotationen

Was uns an grundlegenden Veränderungen einer Darstellung noch fehlt, ist die Rotation des Bildnisses um eine Achse bezeihungsweise um den Ursprung des Koordinatensystemes. Tatsächlich haben wir bereits eine entsprechende Transformation vorgenommen, denn unsere zweite Methode, um einen Kreis auf den Bildschirm zu bringen, tut nichts anderes, als einen (einzigen) Punkt in einer konstanten Entfernung (Radius) um den Koordinatenursprung (Mittelpunkt) zu drehen.

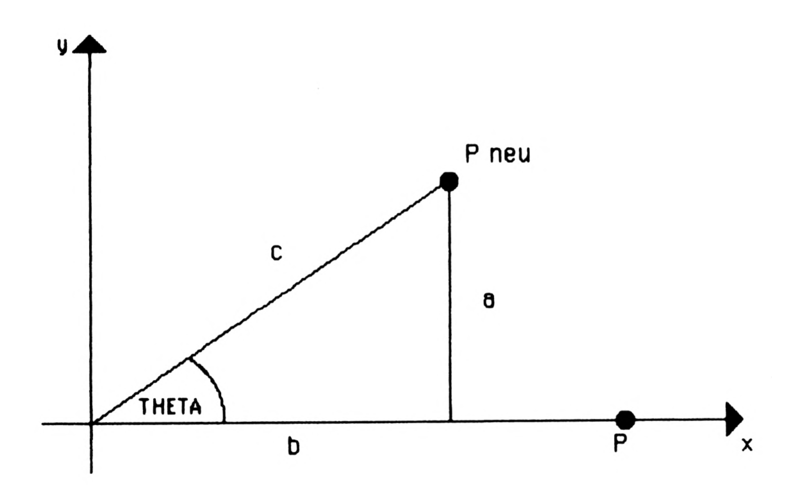
Die Matrix, welche eine solche Rotation repräsentiert, können wir der Trigonometrie entnehmen.

Unter Trigonometrie, oder Drehwinkelmessung, versteht man das Messen und Berechnen von Dreiecken.
Handelt es sich dabei um rechtwinklige Dreiecke, so ist die Berechnung aller Werte aus zwei Teilangaben besonders einfach.
Denn zum einen müssen die Winkel an der Hypotenuse zusammen 90 Grad ergeben, zum anderen gilt der pythagoreische Lehrsatz.





Um nun die Sätze der Trigonometrie anwenden zu können, kann jede durch gerade Linien begrenzte Figur in Dreiecke zerlegt werden, was insbesonders auch für jeden Punkt innerhalb eines kartesischen Koordinatensystemes gilt:



Die trigonometrischen Funktionen SIN() und COS() bilden nun die Verhältnisse der Seiten in einem rechtwenkligen Dreieck. So gilt:

$$\text{SIN(THETA)} = \frac{a}{c}$$

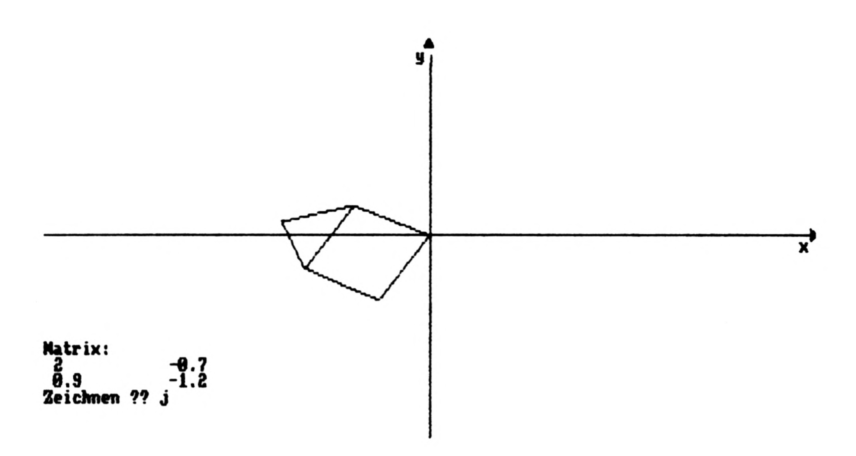
$$\text{COS(THETA)} = \frac{b}{c}$$

THETA ist der gewünschte Drehwinkel, a entspricht einem Teilstück der y-Achse und b einem Teilstück der x-Achse. Unsere Matrix lautet daher:

cos(theta)	sin(theta)
-sin(theta)	cos(theta)

Falls Sie nun mit der Drehung ein wenig experimentieren möchten, so reicht es aus, in den Zeilen 630 und 640 die neuen Transformationsformeln einzusetzen:

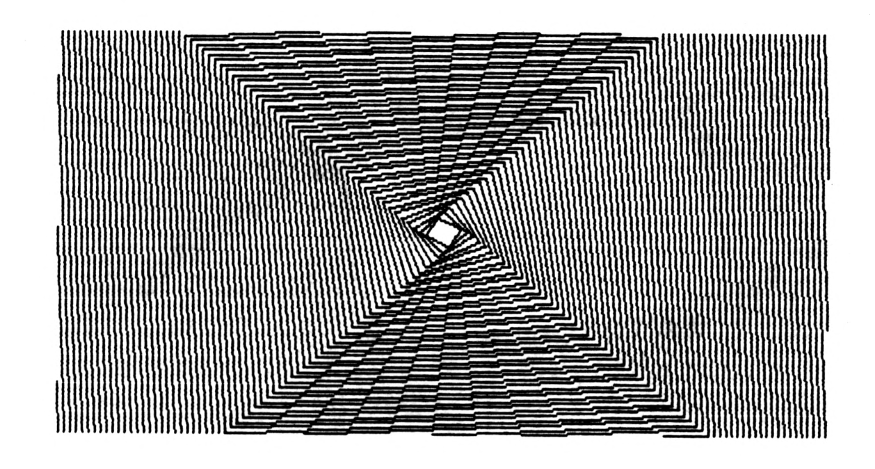
```
630 xneu(punkt)=COS(a)*x(punkt)+(-SIN(c)
)*y(punkt)
640 yneu(punkt)=SIN(b)*x(punkt)+COS(d)*y
(punkt)
```



von Shapes, Matrizen und Koordinatentransformationen lassen sich nicht nur in Programmen zum darstellenden Zeichnen von Objekten (CAD) finden, sondern Shapes und Koordinatentransformationen sind meistens auch das Kernstück eindrucksvoller Computergrafikprogramme.

Die Vorgehensweise sieht dabei so aus, daß ein einfaches Shape, ein Dreieck oder ein Viereck, manchmal auch ein Vieleck, um (s)einen Mittelpunkt gedreht wird:

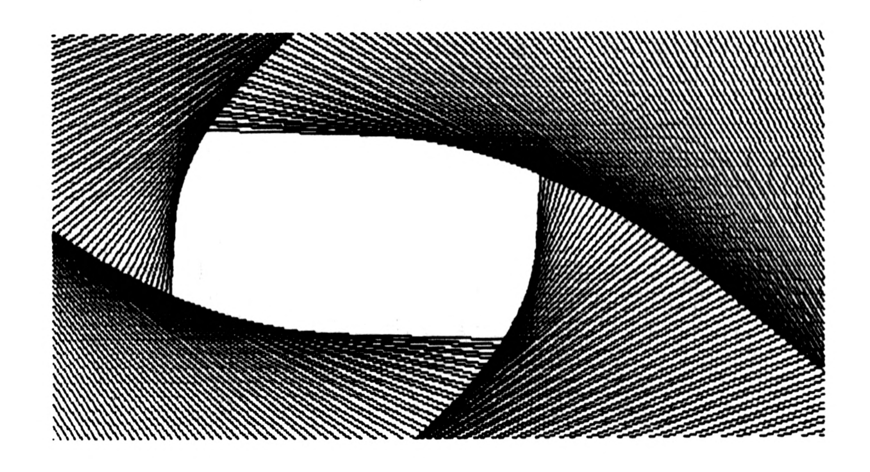
```
10 REM gedrehte figur
20 REM hier viereck
30 REM ---------------
40 MODE 2:PAPER 0:PEN 1:INK 0,12:INK 1,0
:BORDER 12
45 ORIGIN 320,200
50 x1=-220:x2=80:x3=80:x4=-220
60 y1=-100:y2=-100:y3=100:y4=100
70 sw=5
80 FOR anzahl=1 TO 100
90 x=x+sw:y=y+sw
100 x1=x1-sw:x2=x2-sw:x3=x3+sw:x4=x4+sw
110 y1=y1+sw:y2=y2-sw:y3=y3-sw:y4=y4+sw
120 PLOT x1,y1
130 DRAW x2,y2
140 DRAW x3,y3
150 DRAW x4,y4
160 DRAW x1,y1
170 NEXT anzahl
```



Bestimmend für das Aussehen der Grafik sind zum einen selbstverständlich die Originalpunkte (Zeilen 50 und 60), zum anderen aber auch der Koordinatenursprung, die Anzahl der Wiederholungen und auch die Schrittweite.
So führt folgende Änderung

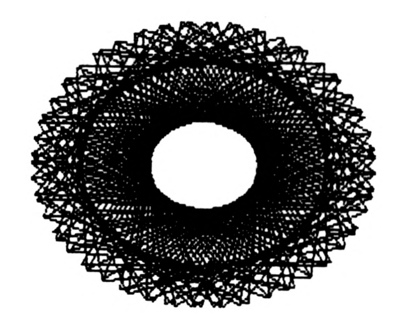
```
50 x1=-220:x2=80:x3=80:x4=-220
60 y1=-100:y2=-100:y3=100:y4=100
```

bereits zu einem anderen Bild.



Nun erweist sich dieses Programm noch als recht starr, werden die Daten doch nicht innerhalb eines Arrays gespeichert, und lassen sich unsere Transformationsformeln auch nicht sogleich entdecken.

Ganz anders sieht es jedoch bei folgendem Programm aus:



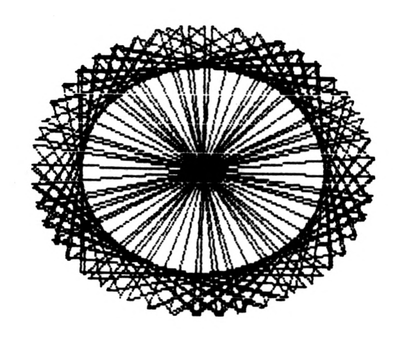
```
10 REM spirograph
20 REM um mp gedrehte vielecke
30 REM ------------------------
40 MODE 2:PAPER 0:PEN 1:INK 1,12:BORDER
12:INK 1,12
50 DEFINT a-z
60 REM ursprung=bildschirmmitte
70 ORIGIN 320,200
80 REM wie viele eckpunkte ?
90 READ anz
100 REM originalpunkte einlesen
110 FOR p=1 TO anz
120 READ xo(p),yo(p)
130 NEXT p
140 REM koordinatentransformation
150 REM wie oft und um wieviel drehen
160 FOR theta= 1 TO 180 STEP 3
170 REM alle eckpunkte umrechnen
180 FOR p=1 TO anz
190 x(p)=xo(p)*COS(theta)-yo(p)*SIN(thet
a)
200 y(p)=xo(p)*SIN(theta)+yo(p)*COS(thet
a)
210 NEXT p
220 REM und zeichnen
230 GOSUB 250
240 NEXT theta:CALL &A000
250 PLOT x(1),y(1):DRAW x(2),y(2):DRAW x
(3),y(3):DRAW x(1),y(1):RETURN
260 REM hier stehen die daten der figur

270 REM zuerst wie viele eckpunkte
280 DATA 3,-100,-100,100,-100,0,100
```

Die Daten des Shapes, welches wir rotieren und skalieren wollen, stehen wie gewohnt am Ende des Programmes; damit die Werte auch richtig eingelesen werden, steht zu Beginn wieder die Anzahl der Punkte.
Gelesen und den Variablen xo und yo zugewiesen werden die Eckpunkte innerhalb der Schleife von 110 bis 130.
Der Drehwinkel schließlich wird in Zeile 160 festgelegt, und in den Zeilen 190 und 200 finden wir auch die Transformationsgleichungen.
Sie erkennen, daß hier eine Drehung durchgeführt wird, was bereits zu ansprechneden Grafiken führt, doch können Sie das Shape darüber hinaus auch noch vergrößern oder verkleinern lassen.

Auch hier machen die Daten wieder das Bild:

```
280 DATA 4,-100,-100,100,-100,100,100,-1
00,100
```



Die vielfältigen Möglichkeiten allein bei der Drehung von Shapes zeigt folgendes Programm in einer unablässigen Show:

```
10 REM computergrafik 2
20 REM um mp gedrehte vielecke
30 REM -----------------------
40 MODE 2:PAPER 0:PEN 1:INK 0,12:BORDER
12:INK 1,0
50 DEFINT a-z
60 DIM xo(15),yo(15),x(15),y(15)
70 REM ursprung=bildschirmmitte
80 ORIGIN 320,200
90 REM wie viele eckpunkte ?
100 anz=RND(1)*15
110 REM originalpunkte einlesen
120 FOR p=1 TO anz
130 xo(p)=RND(1)*320-160:yo(p)=RND(1)*20
0-100
140 NEXT p
150 REM koordinatentransformation
160 REM wie oft und um wieviel drehen
170 thetamax=RND(1)*360:sw=RND(1)*10
180 FOR theta= 1 TO thetamax STEP sw
190 REM alle eckpunkte umrechnen
200 FOR p=1 TO anz
210 x(p)=xo(p)*COS(theta)-yo(p)*SIN(thet
a)
220 y(p)=xo(p)*SIN(theta)+yo(p)*COS(thet
a)
230 NEXT p
240 REM massstab=massstab*1.1
250 REM und zeichnen
260 GOSUB 290
270 NEXT theta
280 FOR i=1 TO 5000:NEXT:CLS:GOTO 100
290 PLOT x(1),y(1):FOR p=1 TO anz:DRAW x
(p),y(p):NEXT p:DRAW x(1),y(1):RETURN
```

Funktionen

Ein anderer wichtiger Aspekt der zweidimensionalen Computergrafik ist das Plotten von Funktionen.
Darunter versteht man die grafische Darstellung der Beziehung zwischen jeweils zwei Werten zueinander, wobei die Art der Beziehung meistens durch eine mathematische Formel angegeben werden kann.

So wurde denn auch der Begriff der Funktion bereits zu Beginn des 18. Jahrhunderts definiert.
Doch erwies sich die damals formulierte Beschreibung einer Funktion als 'veränderliche Größe, die von einer anderen veränderlichen Größe abhängt' als nicht exakt genug für die späteren Ansprüche der Mathematik.
Denn charakteristisch für das Wesen der Funktion sind nicht die unterschiedlichen Größen (Zahlenwerte), als vielmehr die Zuordnung dieser Größen zueinander.
So ist es auch zu erklären, daß nicht nur Zahlen, sondern auch eine Reihe von Gegenständen per Funktion zugeordnet werden können.

Natürlich wird ein Mathematiker niemals von einer Reihe, sondern von einer Menge sprechen, und er wird uns erklären:

"Eine Funktion ordnet jedem Element einer Menge ein bestimmtes Element einer anderen Menge zu."

Als Beispiel könnte hier die Menge der Autos auf der einen, und die Menge der Fahrer auf der anderen Seite betrachtet werden, denn zu jedem Auto läßt sich ein Fahrer nennen.
Leider ist diese recht schöne Beziehung noch keine Funktion, denn jedem Auto kann nicht 'ein bestimmter' Fahrer zugewiesen werden.

Besser ist es da schon, wenn man die Autos in Beziehung zu den augenblicklich eingetragenen Haltern setzt, denn dann kann in Zusammenhang mit jedem Auto auch nur eine einzige Person genannt werden.

Nennen wir die Autos nun x und die Kfz-Halter y, dann läßt sich eine Definition der Funktionen folgendermaßen formulieren:

Eine Funktion ist eine eindeutige Abbildung zweier Mengen, bzw. einer gewissen Menge von geordneten Paaren (x,y), für die gilt, daß es zu jedem x genau ein y gibt.

Symbolisch wird der Zusammenhang als

$$y = f(x)$$

dargestellt, wobei x den Elementen der ersten Menge entspricht, eben allen den Elementen, für die die Funktion definiert ist, und y stellvertretend für die errechneten Werte steht.
Analog dazu spricht man auch von Definitions- und Wertebereich.

Darstellung von Funktionen

Jedem Schüler werden heute zumindest drei Methoden zur Darstellung von Funktionen nahebebracht.

Schnell wieder vergessen wird der Funktionsgraph, bei dem der Definitionsbereich und der Wertebereich in Form von Kreisen oder Ovalen dargestellt, und die Zuordnungen in Form von Pfeilen veranschaulicht werden.
Eine Funktion erkennt man bei dieser Darstellungsform dann daran, daß von jedem Element der Definitionsmenge nur ein einziger Pfeil ausgeht; dennoch können mehrere Pfeilspitzen auf ein Element der Ergebnismenge weisen.

Nachdem solch ein Graph dann erstellt war, ließ sich aus ihm leicht eine Wertetabelle erstellen.:
die einzelnen Elemente des Definitionsbereiches wurden neben- oder untereinander notiert, und der per Pfeil zugeordnete Wert dann dazugeschrieben.

In vielen Fällen ließ sich dann jedem Zahlenpaar dieser Wertetafel ein Punkt P in der Ebene zuordnen, wodurch nach und nach ein Bild der Funktion entstand.

Üblich in den meisten Fällen war dabei die Verwendung des kartesischen Koordinatensystems, wobei die x-Werte auf der waagrechten und die durch die Funktionsgleichung festgelegten Werte auf der senkrechten Achse abgetragen wurden.

Abhängig von der Beschaffenheit des Definitionsbereiches und der Funktionsgleichung erhielt man dann eine Reihe von Punkten, Kurvenstücken oder auch eine ausgezeichnete Funktionskurve.

Funktionenplotter

Alles was wir im folgenden tun müssen, ist, diese Technik in ein Programm umzusetzen und dabei die Gegebenheiten des Computers zu berücksichtigen.

Der erste Schritt wäre demnach die Erstellung einer Wertetafel:

```
10 PAPER 0:PEN 1:BORDER 12:INK 0,12:INK 1
,0
200 FOR x=1 TO 10
210 y=x*x
220 PRINT x;y
230 NEXT x
```

In Zeile 200 legen wir unseren Definitionsbereich fest, in diesem Beispiel von 1 bis 10, in Zeile 210 steht die Funktion und wird f(x) berechnet, und Zeile 220 ist dafür verantwortlich, daß sämtliche Werte in Form einer Tabelle untereinander ausgegeben werden.

Auch eine einfache Skizzierung der Funktion ist bereits möglich; ersetzen Sie

1. Zeile 220 durch PRINT TAB(y)"*"

und ergänzen Sie

2. 20 MODE 2

Nach dem Start des kurzen Programmes werden Sie den berechneten Ast der Normalparabel ohne Schwierigkeiten erkennen.

Ähnlich wären Sie auch bei einer Abbildung der Funktion auf Papier vorgegangen.
Sie hätten nämlich ein rechtwinkliges Koordinatensystem gewählt bei dem die x-Werte von links nach rechts an-, und die y-Werte von unten nach oben aufsteigen.
Sie hätten zunächst die Koordinatenachsen, und daran anschließend die berechneten Punkte gezeichnet.
Schließlich hätten Sie alle so abgebildeten Punkte mit einer geschwungenen Linie untereinander verbunden.

Wie wir bereits festgestellt haben, verhält der CPC sich in dieser Beziehung direkt menschlich, orientiert es sich auf seiner Zeichenfläche doch ebenso und erlaubt er sogar die freie Wahl des Koordinatenursprungs.

Ein Koordinatenkreuz hatten wir ja früher schon gezeichnet, und auch die Bezeichnungen x und y sind uns geläufig - also sollte es doch reichen, das Programm um die fühere Routine zu ergänzen und die PRINT-Anweisung in 220 durch ein PLOT (x,y) zu ersetzen:

```
10 PAPER 0:PEN 1:BORDER 12:INK 0,12:INK
1,0
20 MODE 2
100 ORIGIN 320,200:CLG:TAG
110 PLOT -320,0:DRAW 320,0
120 MOVER -5,6:PRINT CHR$(246);
130 MOVER -17,-10:PRINT"x";
140 PLOT 0,-200:DRAW 0,200
150 MOVER -4,-2:PRINT CHR$(244);
160 MOVER -17,-10:PRINT"y";
170 TAGOFF
200 FOR x=-10 TO 10
210 y=x*x
220 PLOT x,y
230 NEXT x
```

Tatsächlich erscheint die Parabel korrekt auf dem Schirm, nur die Größe des Abbildes entspricht nicht so ganz unseren Erwartungen.
Doch ist das für uns inzwischen kein Problem mehr, denn daß sich Skalierungen durch Multiplikation mit einem konstanten Faktor bewältigen lassen, haben wir inzwischen gelernt.

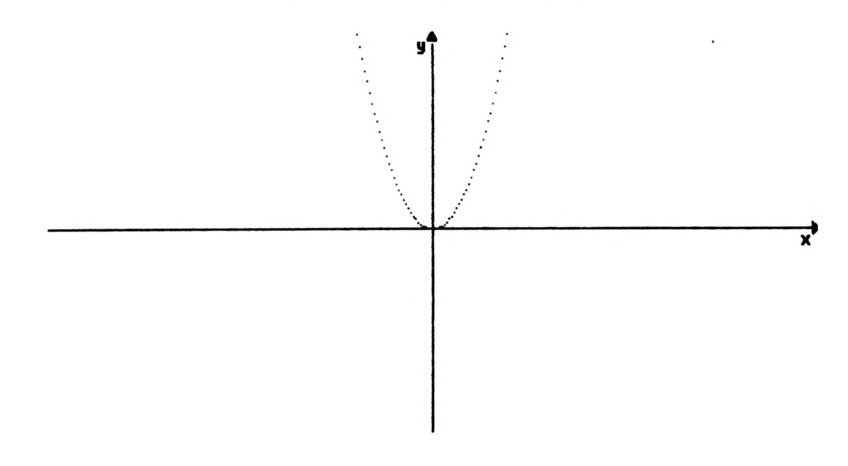
Lassen wir die Kurve doch vergrößert zeichnen, und legen wir einen Maßstabsfaktor von zunächst einmal 20 fest:

```
180 mf=20
220 PLOT x*mf,y*mf
```

Aus dem erzeugten Bild läßt sich annehmen, daß die Lage der gesetzten Punkte annähernd richtig ist, doch muß man eine Funktion schon recht gut kennen, um solche Schlüsse aus einem Minimum an Punkten ziehen zu können.

Was noch fehlt, sind einfach mehr Punkte - um nicht zu sagen möglichst viele.

Ergänzen wir Zeile 200 um STEP .1, und schon:



Um nun einen geschlossenen Linienzug zu erhalten, stehen uns zwei Wege offen: wir könnten auch die zwischen den bereits gezeichneten Punkten liegenden Punkte berechnen, oder wir lassen den Computer unsere eigene Arbeitsweise übernehmen und ihn eine Verbindungslinie von Punkt zu Punkt ziehen.

Bei Anwendung des letzten Verfahrens ist unser Programm zwar schnell, aber leider ungenau, denn der Computer kann die Krümmung der Kurve nicht abschätzen wie wir es tun, sondern zeichnet immer eine Gerade.

Und das andere Lösungsverfahren ist zwar exakt, doch benötigt das Programm bei ausreichend kleinen Schrittweiten aufgrund der enormen Rechenarbeit eben auch entsprechend viel Zeit.
Überzeugen Sie sich davon, indem Sie die Zeilen 200 und 210 austauschen:

```
200 FOR x=-10 TO 10 STEP 0.01
210 y=SIN(x)
```

Doch wie wäre es mit einer Kombination beider Vorschläge und darüber hinaus mit der Eingabe sämtlicher Kontrollwerte nach Wunsch, so daß wir eine Funktion beliebig vergrößern können falls ihr Abbild uns nicht genug Einzelheiten preisgibt ?

Geben Sie das folgende Listing ein, und experimentieren Sie mit dem Programm.
Vergessen Sie dabei aber auch nicht, andere Funktionen als die Normalparabel in Programmzeile 10000 einzusetzen.

```
10 PAPER 0:PEN 1:BORDER 12:INK 0,12:INK
1,0
20 MODE 2
100 REM achsenkreuz
110 ORIGIN 320,200:CLG:TAG
120 PLOT -320,0:DRAW 320,0
130 MOVER -5,6:PRINT CHR$(246);
140 MOVER -17,-10:PRINT"x";
150 PLOT 0,-200:DRAW 0,200
160 MOVER -4,-2:PRINT CHR$(244);
170 MOVER -17,-10:PRINT"y";
180 TAGOFF
190 REM eingabe plotdaten
200 INPUT"von x= ";x:xmin=x:GOSUB 10000:
ymin=x:INPUT"bis x= ";x:xmax=x:GOSUB 100
00:ymax=x:INPUT"Genauigkeit ";genauigkei
t
210 INPUT"Vergroesserungsfaktor ";mf
300 REM funktion plotten
310 x=xmin:GOSUB 10000:PLOT x*mf,y*mf:RE
M erster Punkt der Kurve
320 FOR x=xmin TO xmax STEP genauigkeit

330 GOSUB 10000
340 DRAW x*mf,y*mf
350 NEXT x
360 GOSUB 410:REM skalierung
370 END
400 REM skalaierung x-achse
410 FOR x=xmin TO xmax STEP 10
420 GOSUB 10000
430 PLOT x*mf,5:DRAW x*mf,-5
440 MOVE 0,-10:PRINT INT(x);
450 NEXT x
460 RETURN
9990 REM darzustellende Funktion in 1000
0 einsetzen
10000 y=SIN(x)
10010 RETURN
```

Eigentlich funktioniert doch nun schon alles recht ordentlich. Nur eine Beschriftung der Achsen fehlt noch, um das Bild aussagekräftiger zu machen.

Dazu gibt es wiederum mehrer Techniken, von denen eine hier vorgestellt werden soll.
So kann zunächst festgelegt werden, mit wie vielen Teilstrichen jede Achse markiert werden sollen.
In unserem Fall sollen es jeweils zehn sein, was bedeutet, daß sich auf der x-Achse alle 64 x-Einheiten eine Markierung befinden muß. Für die y-Achse beträgt der Abstand von Markierung zu Markierung demgemäß 40 Einheiten.

Nach dieser Vorüberlegung kann eine Schleife aufgebaut werden, welche die Skalierung vornimmt.
Dabei wird der Zahlenwert einer jeden Markierung der Quotient aus der Anzahl der Zeicheneinheiten durch den Maßstabsfaktor sein.
Wurde der Wert so errechet, kann die Beschriftung nach TAG mittels des Grafik-Cursors vorgenommen werden.
Allerdings sollte zuvor nicht vergessen werden, die Betriebsart 'ODER-Verknüpfung' einzuschalten, so daß nicht Teile des Funktionsbildes gelöscht werden.

Ein letztes Problem kann eine bei der Berechnung der Funktionswerte auftretende Division durch Null sein.
Dieser Fehler würde normalerweise zum Programmabbruch führen. Deshalb wird innerhalb von Plotprogrammen entweder zu jedem Wert des Definitionsbereiches ein konstanter Faktor (vielleicht 0.00001) addiert, oder das Programm wird angewiesen, bei Auftreten eines solchen Fehlers sogleich mit dem nächsten Wert weiterzuarbeiten.

```
5 REM plotter
10 ON ERROR GOTO 20000:REM im falle eine
s falles
20 PAPER 0:PEN 1:BORDER 12:INK 0,12:INK
1,0
30 MODE 2
50 REM eingabe plotdaten
60 INPUT"von x= ";x:xmin=x:GOSUB 10000:y
min=x:INPUT"bis x= ";x:xmax=x:GOSUB 1000
0:ymax=x:INPUT"Genauigkeit ";genauigkeit

70 INPUT"Vergroesserungsfaktor ";mf

90 MODE 2
100 REM achsenkreuz
110 ORIGIN 320,200:CLG:TAG
120 PLOT -320,0:DRAW 320,0
130 MOVER -5,6:PRINT CHR$(246);
140 MOVER -17,-10:PRINT"x";
150 PLOT 0,-200:DRAW 0,200
160 MOVER -4,-2:PRINT CHR$(244);
170 MOVER -17,-10:PRINT"y";
180 TAGOFF
300 REM funktion plotten
310 x=xmin:GOSUB 10000:PLOT x*mf,y*mf:RE
M erster Punkt der Kurve
320 FOR x=xmin TO xmax STEP genauigkeit

330 GOSUB 10000
340 DRAW x*mf,y*mf
350 NEXT x
360 GOSUB 400:REM skalierung
370 IF INKEY$="" THEN 370 ELSE 20
390 REM skalierung x-achse
400 PRINT CHR$(23);CHR$(1);:TAG:REM uebe
rschreiben ohne loeschen
410 FOR xa=-320 TO 320 STEP 64
```

```
420 x=xa/mf:GOSUB 10000
430 PLOT xa,5:DRAW xa,-5
440 MOVER -19,-10:PRINT x;
450 NEXT xa
490 REM skalierung y-achse
500 FOR ya=40 TO 200 STEP 40
510 y=ya/mf
520 PLOT -5,ya:DRAW 5,ya
530 MOVER -1,6:PRINT y;
540 NEXT ya
550 REM -y -Achse
560 FOR ya=-40 TO -200 STEP -40
570 y=ya/mf
580 PLOT -5,ya:DRAW 5,ya
590 MOVER -1,6:PRINT y;
600 NEXT ya
610 TAGOFF
620 PRINT CHR$(23);CHR$(0):REM ueberschr
eiben aus
630 RETURN
9990 REM darzustellende Funktion in 1000
0 einsetzen
10000 y=SIN(x)/x
10010 RETURN
20000 RESUME NEXT:REM einfach weitermach
en
```

5. KAPITEL

3D - GRAFIK

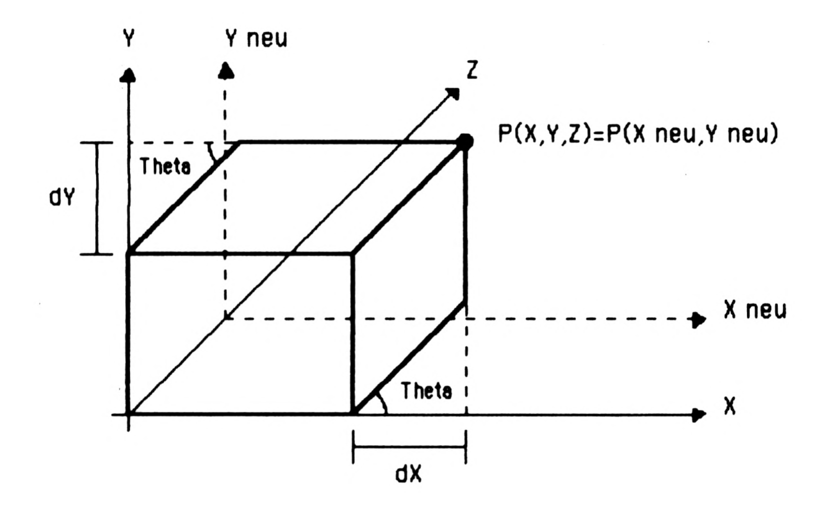
Die dritte Dimension

Zur dreidimensionalen Darstellung von Objekten haben sich im Laufe der Jahre eine Reihe von Verfahren entwickelt, die sich zum Teil auch auf Computern einsetzen lassen.

Doch handelt es sich meist nur um die Illusion eines dreidimensionalen Gebildes, denn wie sollte eine zweidimensionale Zeichenfläche in die Tiefe gehen können ?
Und so sieht man in den meisten Fällen auch nur perspektivische Abbildungen anstelle von wirklichkeitsgetreuen 3d-Bildern.

Perspektivische Abbildungen

Bei dieser Art von 3d-Bildern wie wir sie täglich im Fernsehen und auf Fotos sehen, ist nämlich nicht zusätzlicher technischer Aufwand erforderlich, sondern nur ein wenig angwewandte Mathematik.
Denn bei dieser 'optischen Täuschung' werden beispielsweise sämtliche Eckpunkte eines abzubildenden Hauses auf eine zweidimensionale Fläche projiziert.



Dabei werden die Punkte der dritten Achse, der z- oder Raumachse entlang der x-Achse abgebildet.
Die Zeichnung macht deutlich, daß zur Darstellung des hinteren rechten Punktes des Würfels eine Koordinatentransformation vorgenommen wurde, und zwar um den Betrag, um den P(x,y,z) von der x-Achse entfernt war, also genau um z.
Im neuen Koordinatensystem xneu,yneu läßt der Punkt P(x,y,z) sich dann mit nur zwei Koordinaten angeben: P(xneu,yneu).

Bei einer weiteren Betrachtung der Skizze fällt auf, daß wir mit unserem Wissen bereits in der Lage sein müßten, solche räumlichen Bilder zu erzeugen.
Denn wir haben bereits im vorherigen Kapitel eine Formel kennengelernt, die sich auch in diesem Falle anwenden läßt.

So wird die Lage der Raumachse in Bezug auf die x-Achse konstant sein und außerdem von uns gleich zu Beginn festgelegt werden, der Winkel den die beiden Achsen somit bilden ist also bekannt.

Ebenfalls bekannt ist die Länge der Strecke x, und so können wir die Lage des neuen Bildpunktes P(x,y) von P(x,y,z) mit

x = X2 + z * COS(THETA)

und y = Y2 + z * SIN(THETA)

angeben.

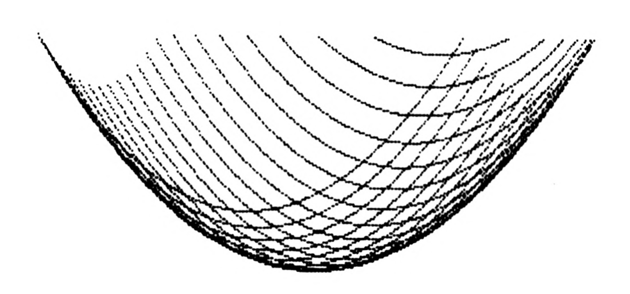
3d - Funktionsplotter

Von der Richtigkeit der vorangegangenen Behauptung können wir uns recht schnell anhand unseres Funktionenplotters überzeugen.
Dort wurde eine Variable in Abhängigkeit von einer anderen dargestellt, nun wollen wir die Variable z in Abhängigkeit von x und y berechnen.

Also werden wir um die erste Schleife herum eine zweite aufbauen, welche dann für die verschiedenen y-Werte verantwortlich sein wird.
Außerdem sehen wir für jede der drei Koordinaten einen eigenen Maßstabsfaktor vor, so daß das Bild der Funktion in alle drei Richtungen des Raumes vergrößert und auch verkleinert werden kann:

```
10 PAPER 0:PEN 1:BORDER 12:INK 0,12:INK
1,0
20 MODE 2
30 ORIGIN 320,200
60 MFX=30:MFY=30:MFZ=5
70 FOR Y=-6 TO 6 STEP 0.5
80 FOR X=-6 TO 6 STEP 0.05
90 GOSUB 10000
100 SX=X*MFX+Y*MFY*COS(45)
110 SY=Z*MFZ+Y*MFY*SIN(45)
120 PLOT SX,SY
130 NEXT X
140 NEXT Y
150 END
10000 Z=X*X+Y*Y
10010 RETURN
```

In den Zeilen 100 und 110, in denen die Schirmkoordinaten berechnet werden, erkennen Sie ohne Schwierigkeiten die angegebenen Transformationsformeln.
In Zeile 10000 wurde als Funktion wieder die Normalparabel eingesetzt, diesmal allerding in räumlicher Form.
Es dürfte bei einem Lauf des Programmes somit eine Abbildung ähnlich der Rundung eines Reagenzglases entstehen:



Wie dicht die einzelnen Linien beieinander liegen, entscheiden Sie mit den Schrittweiten in den Zeilen 70 und 80.
Die Größe der Abbildung können Sie durch verändern der Werte in 60 festlegen.

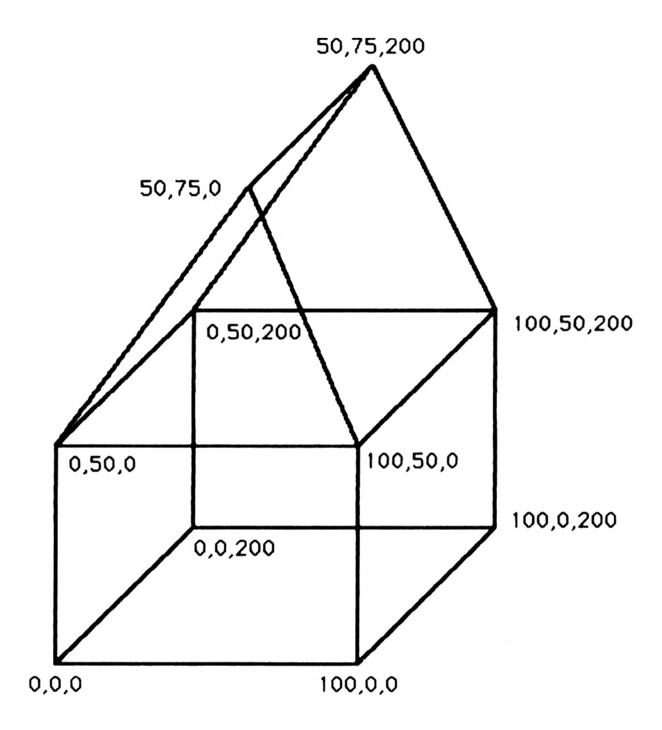
Außerdem sollten Sie obiges Programm noch einmal in Hinblick auf Effizienz untersuchen.
Denn der Winkel, dessen Sinus und Cosinus wieder und wieder berechnet wird, legt die Lage der Raumachse fest und bleibt, zumindest während eines Programmlaufes, konstant.
Somit werden die trigonometrischen Berechnungen auch immer zum gleichen Ergebnis führen, so daß in den Zeilen 100 und 110 auch gleich eine Konstante eingesetzt werden kann.

```
10 REM plot 3d
20 PAPER 0:PEN 1:BORDER 12:INK 0,12:INK
1,0
30 MODE 2
40 ORIGIN 320,200
50 REM masstabsfaktoren fuer alle raumac
hsen
60 INPUT"Faktor x ";mfx:INPUT"faktor y";
mfy:INPUT"faktor z ";mfz
70 REM winkel der raumachse
80 w1=COS(45):w2=SIN(45)
90 REM funktion plotten
100 FOR y=-4 TO 4 STEP 0.25
110 FOR x=-4 TO 4 STEP 0.01
120 GOSUB 10000
130 REM schirmkkoordinaten berechnen
140 sx=x*mfx+y*mfy*w1
150 sy=z*mfz+y*mfy*w2
160 PLOT sx,sy
170 NEXT x
180 NEXT y
190 END
9990 REM funktion in zeile 10000 einsetz
en
10000 z=x*x+y*y
10010 RETURN
```

Naturgemäß besteht der nächste Schritt darin, auf diese Art und Weise auch reale Objekte auf dem Bildschirm darzustellen.
In der Tat ist dies mit unseren Formeln auch möglich, doch wird der abzubildende Gegenstand zunächst erst einmal in ein Zahlenmodell umgesetzt werden müssen, welches dann die Basis für sämtliche Manipulationen darstellt.

Dieser Vorgang könnte so aussehen, daß wir das abzubildende Objekt ausmessen, wobei wir eine Ecke dieses Gegenstandes willkürlich als Nullpunkt bestimmen, und alle anderen markanten Punkte als gemessene Entfernungen zu diesem Punkt angeben.

Unser schon häufig für Manipulationen ähnlicher Art herangezogenes Haus ließe sich dann folgendermaßen beschreiben:



Diese Koordinatentripel müssen dem Rechner nun in irgendeiner Weise zur Bearbeitung angeboten werden, wobei allerdings eine genaue Regelung zu treffen ist, wie die Beziehungen der Punkte untereinander sind.
Denn nur die Punkte allein machen noch kein Bild, erst die sie verbindenden Linien lassen uns die Figur erkennen.

Zweckmäßig wäre es daher, von jeder Linie den Anfangs- und Endpunkt anzugeben, denn dadurch erübrigt sich auch die Festlegung einer bestimmten Reihenfolge des Zeichenvorganges.

Außerdem wird das Bild sich leicht editieren lassen, wenn jede Linie für sich verlängert, verkürzt, verschoben oder gelöscht werden kann.

Aus siebzehn Linien besteht das Haus, folglich kann unser Zahlenmodell wie folgt aussehen:

```
1010 DATA 0,0,0,10,0,0
1020 DATA 10,0,0,10,5,0
1030 DATA 10,5,0,0,5,0
1040 DATA 0,5,0,0,0,0
1050 DATA 10,0,20,10,5,20
1060 DATA 10,5,20,0,5,20
1070 DATA 0,5,20,0,0,20
1080 DATA 0,0,20,10,0,20
1090 DATA 10,0,0,10,0,20
1100 DATA 0,0,0,0,0,20
1110 DATA 0,5,0,0,5,20
1120 DATA 10,5,0,10,5,20
1130 DATA 10,5,0,5,7,0
1140 DATA 0,5,0,5,7,0
1150 DATA 5,7,0,5,7,20
1160 DATA 5,7,20,0,5,20
1170 DATA 5,7,20,10,5,20
```

Wenn wir dann noch zuallererst die Anzahl der Linien angeben, können alle Angaben in eine Variablentabelle der Form xvon,yvon,zvon und xnach,ynach,znach eingelesen werden.

```
100 FOR L=1 TO 17
110 READ XV(L),YV(L),ZV(L),XN(L),YN(L),Z
N(L)
120 NEXT L
```

Damit befindet sich ein naturgetreues Abbild des Objektes im Speicher, allerdings ist es ebenso naturgetreu dreidimensional.
Bevor es gezeichnet werden kann, werden die Werte daher zunächst auf unsere Bildschirmkoordinaten x und y umgerechnet werden müssen.

Das kann mit den bekannten Transformationsgleichungen geschehen:

```
200 FOR L=1 TO 17
210 SXV(L)=XV(L)*ZV(L)*COS(45):SXN(L)=XN
(L)+ZN(L)*COS(45)
220 SYV(L)=YV(L)+ZV(L)*SIN(45):SYN(L)=YN
(L)+ZN(L)*SIN(45)
230 NEXT L
```

Erst dann kann uns nichts mehr daran hindern, das Bild, Linie für Linie, zu zeichnen:

```
300 ORIGIN 320,200:FOR L=1 TO 17
310 PLOT SXV(L),SYV(L):DRAW SXN(L),SYN(L
)
```

```
320 NEXT L
```

Ausblick

Selbstverständlich können die Punkte, sobald sie sich erst einmal im Rechner befinden, nach sämtlichen Regeln der Kunst manipuliert werden.

Im Prinzip ist auch dieses Haus nichts weiter als ein Shape, es wurde nur in etwas anderer Form dargestellt, als in Kapitel 4.
Doch besteht der einzige Unterschied darin, daß dort jeder Punkt, mit Ausnahme des ersten, der Endpunkt einer Linie war, und hier wird zunächst immer erst ihr Beginn angegeben.

Der Umgang mit den Shapes bleibt jedoch gleich, was folgendes kleines Programm, welches schon ein kleines CAD-Programm darstellt, noch einmal in Hinblick auf Vergößerungen, Verkleinerungen und Verzerrungen zeigen soll.

```
10 REM caddemo
20 MODE 2:ORIGIN 320,200
30 mfx=5:mfy=5:mfz=5:w=45
40 ORIGIN 320,200
50 REM wie viele linien ?
60 READ anz
70 REM tabellen fuer koordinaten
80 DIM sxv(anz),sxn(anz),syn(anz),syv(an
z),xv(anz),yv(anz),zv(anz),xn(anz),yn(an
z),zn(anz)
90 REM punkte einlesen
100 FOR l=1 TO anz
110 READ xv(l),yv(l),zv(l),xn(l),yn(l),z
n(l)
120 NEXT l
190 REM in Bildschirmkoordinaten umrechn
en
200 FOR l=1 TO anz
210 sxv(l)=e+xv(l)*mfx+zv(l)*COS(w)*mfz:
sxn(l)=xn(l)*mfx+zn(l)*COS(w)*mfz
220 syv(l)=yv(l)*mfy+zv(l)*SIN(w)*mfz:sy
n(l)=yn(l)*mfy+zn(l)*SIN(w)*mfz
230 NEXT l:e=0
240 GOTO 400
290 REM bild zeichnen
300 ORIGIN 320,200:FOR l=1 TO anz
310 PLOT sxv(l),syv(l):DRAW sxn(l),syn(l
)
320 NEXT l
330 IF INKEY$="" THEN 330 ELSE 400
390 REM menue
400 CLS:PRINT"Sie wuenschen:":PRINT:PRIN
T
410 PRINT"1 - vergroessern/verkleinern
420 PRINT"2 - verzerren
430 PRINT"3 - neuen Blickwinkel
440 PRINT"4 - Originalbild"
```

```
450 PRINT"5 - Bild zeigen"
460 PRINT"6 - ENDE"
470 a$=INKEY$:IF a$="" THEN 470
480 a=VAL(a$)
490 IF (a<0 OR a>6) THEN 400
500 ON a GOTO 600,700,800,820,900,910
590 REM massstab
600 CLS:INPUT"Geben Sie nun fuer eine Ve
rgroesserung eine Zahl groesser 1, und f
uer eine        Verkleinerung eine Zahl zw
ischen 0 und 1 ein. ";mfg
610 mfx=mfg:mfy=mfg:mfz=mfg
620 GOTO 200
690 REM verzerren
700 CLS:INPUT"in welche Richtung: x=1   y
=2  z=3 ";r:INPUT"Geben Sie nun fuer ein
e Streckung eine Zahl groesser 1, und fu
er eine Stauchung eine Zahl zwischen 0 u
nd 1 ein. ";mf
710 IF (r<0 OR r>3) THEN 700
720 ON r GOTO 730,740,750
730 FOR l=1 TO anz:xv(l)=xv(l)*mf:xn(l)=
xn(l)*mf:NEXT l:GOTO 200
740 FOR l=1 TO anz:yv(l)=yv(l)*mf:yn(l)=
yn(l)*mf:NEXT l:GOTO 200
750 FOR l=1 TO anz:zv(l)=zv(l)*mf:zn(l)=
zn(l)*mf:NEXT l:GOTO 200
790 REM raumachse
800 CLS:INPUT"Wieviel Grad liegen zwisch
en der x- und der z-Achse ";w
810 GOTO 200
820 CLS:RUN
890 REM bild zeigen
900 MODE 2:GOTO 300
910 CLS:END
980 REM daten des objektes
990 REM schema: startpunkt linie (x,y,z)
```

```
 - endpunkt linie (x,y,z)
1000 DATA 17
1010 DATA 0,0,0,10,0,0
1020 DATA 10,0,0,10,5,0
1030 DATA 10,5,0,0,5,0
1040 DATA 0,5,0,0,0,0
1050 DATA 10,0,20,10,5,20
1060 DATA 10,5,20,0,5,20
1070 DATA 0,5,20,0,0,20
1080 DATA 0,0,20,10,0,20
1090 DATA 10,0,0,10,0,20
1100 DATA 0,0,0,0,0,20
1110 DATA 0,5,0,0,5,20
1120 DATA 10,5,0,10,5,20
1130 DATA 10,5,0,5,7,0
1140 DATA 0,5,0,5,7,0
1150 DATA 5,7,0,5,7,20
1160 DATA 5,7,20,0,5,20
1170 DATA 5,7,20,10,5,20
```

6. KAPITEL
SOUND

Der Ton macht die Musik

ist nicht nur eine bloße Redensart, sondern diese Redewendung trifft, wie wir noch sehen werden, gerade im Bereich der Musikerzeugung mittels elektronischer Hilfsmittel mehr zu als Sie vielleicht glauben.

Wie wohl jeder weiß, 'besteht' ein Ton aus Schwingungen; aus Schwingungen, die durch die Luft übertragen schließlich an unser Ohr gelangen, und dort das Trommelfell in mehr oder weniger schnelle Bewegungen versetzen.
Diese Schwingungen, in elekrische Impulse verwandelt, lassen in unserem Gehirn den Eindruck eines Geräusches entstehen. Dabei empfinden wir den Ton umso höher, desto schneller das Trommelfell vibriert.

Doch wie läßt es sich erklären, daß eine Saite, welche in einem Klavier pro Sekunde 440 mal hin und herschwingt, anders klingt, als eine gleich lange Saite in einem Flügel ?

Ganz klar, wird vielleicht jemand antworten, das eine Instrument hat einen weichen, und das andere einen harten Anschlag.

Doch wie wirkt sich dieser unterschiedliche Anschlag beispielsweise auf den Ton aus ?

Um diese Frage beantworten zu können, muß man sich zunächst über den Aufbau eines Tones klar werden.
Unterscheiden lassen sich im wesentlichen die regelmäßigen, periodischen Klänge von den unregelmäßigen, aperiodischen Geräuschen.

Schon die Unterscheidung in Klang auf der einen, und Geräusch auf der anderen Seite weist auf die Unterschiede in iher Erscheinungsform hin; als Beispiele lassen sich die von einer Blockflöte erzeugten Töne, wie auch die vom Brausen

des Windes erzeugten Geräusche nennen.

Doch ist in der Regel kein von einem Instrument erzeugter Ton absolut rein, sondern es handelt sich immer um ein Mischmasch aus mehreren Tönen, aus Grundschwingung und Oberwellen.

Diese Zusammenhänge hat zu Beginn des 19. Jahrhunderts der französische Mathematiker Joseph de Fourier in seinem Werk 'Theorie analytique de la chaleur' klargestellt.
Er machte sich daran, periodische Funktionen durch Reihen von periodischen Funktionen darzustellen, und mit seinen Theorien schuf er die Voraussetzungen für viele Anwendungen in Mathematik, Physik und Technik.

So entwickelte er das Verfahren der 'harmonischen Analyse von Klängen', bei der eine Schwingung in eine Reihe von reinen Sinusschwingungen und einen konstanten Anteil zerlegt werden.
Er stellte fest, daß die neben der Grundschwingung auftretenden Oberschwingungen immer als ein Vielfaches dieser Grundfrequenz auftreten.

Unter Berücksichtigung dieser Fourierreihenentwicklung ist natürlich auch die Umkehrung des Vorganges, die harmonische Synthese, möglich.
Dabei werden die einzelnen Schwingungen addiert, und es ergibt sich eine Resultierende.
Auf diese Weise lassen sich andere Schwingungsformen konstruieren, wie zum Beispiel die Rechteck oder die Sägezahnschwingung.

Neben der Frequenz der Grundschwingung, die die Höhe des Tones festlegt, entscheidet der Oberwellengehalt, die Form der Kurve, über den Klang des Tones.

Der Synthesizer

Dieses Wissen zunutze gemacht hat sich Bob Moog, der im Jahre 1964 seinen Moog-Synthesizer vorstellte.
Er konstruierte damals ein technisches Ungetüm, das es ihm erlaubte, die klangbestimmenden Glieder eines Tones, Frequenz, Oberwellengehalt und Lautstärke, nach Wunsch zu verändern.

Runde zwanzig Jahre später war es dann soweit, daß ein einziges IC, digital gesteuert, zu ähnlichen, und teilweise sogar besseren, Leistungen fähig ist, wie die damaligen Moog-Synthesizer.

So besitzt der in den CPC-464 eingesetzte Chip AY-3-8912 von General Instruments drei Stimmen, was zwar einerseits nur einem dreifingrigem Klavierspieler entspricht, andererseits aber auch schon zu ganz ordentlichen Ergebnissen führen kann.

Darüber hinaus sind die Kanäle so nach außen auf die Anschlußbuchse gelegt worden, daß sich bei Verwendung einer angeschlossenen Stereoanlage auch Stereoeffekte ergeben. So werden Sie die auf dem Kanal A erzeugten Töne nur links, die von C nur rechts, und die von B in der Mitte, also auf beiden Kanälen hören.

Die Lautstärke läßt sich in bis zu 15 Stufen regeln, die Länge der Note ist auf ebenso einfache Weise steuerbar, der Klang, wie auch die Frequenz, können verändert werden - und so weiter und so fort!

Der Ton - SOUND

Die grundsätzlichen und wichtigsten Einstellungen nehmen Sie mittels der Soundanweisung vor.

Darunter fallen:

- Wahl des Tongenerators
- Wahl der Note
- Bestimmung der Spieldauer
- Festlegung der Lautstärke

Außerdem können Sie jedem Ton eine

- Lautstärken Hüllkurve
- Ton Hüllkurve
- Geräusch-Periode

zuordnen.

Die Kanal-Wahl

im Handbuch immer als Kanal-Status bezeichnet, nehmen Sie mit dem ersten Parameter des Sound-Befehles vor.
Lautet Ihre Anweisung SOUND 1, ..., so hören Sie Kanal A, bei SOUND 2... erklingt Kanal B, und SOUND 3, ... schließlich, bringt Kanal C zum Klingen.

Häufig jedoch werden Sie sich wünschen, mehrere Kanäle gleichzeitig zum Klingen zu bringen, was ganz einfach dadurch geschehen kann, daß Sie die Werte der gewünschten Kanäle addieren und dann die Summe einsetzen.
So bedeutet SOUND 5,... daß die im folgenden angegebene Note gleichzeitig auf Kanal B und C gespielt wird, und SOUND 3,... bedeutet Kanal A und B.

So erklärt sich auch, daß die drei Kanäle der Reihe nach nicht einfach mit SOUND 1..., SOUND 2... und SOUND 3 angesprochen werden können.

Außerdem können Sie mit dem ersten Wert des Sound-Kommandos die Kanäle synchronisieren, wozu Ihnen die Rendezvous-Technik zur Verfügung steht, wie auch das Setzen von Haltepunkte möglich ist, die später mit RELEASE wieder gelöst werden müssen.

SOUND	Auswirkung
1	nur Kanal A
2	nur Kanal B
4	nur Kanal C
8	Rendezvous mit Kanal A
16	Rendezvous mit Kanal B
32	Rendezvous mit Kanal C
64	Halten
128	Rauschen

Die Frequenz

wird über den Wert Ton-Periode angegeben. Wenn Sie Noten in SOUND-Anweisungen umsetzen wollen, so können Sie die Periode einer jeden Note im Anhang des Handbuches ablesen, und diesen Wert an zweiter Stelle des Sound-Befehles einsetzen. Berechnet wird die Frequenz ungefähr mit 125000/Periode.

Sollte in Ihnen nun das Verlangen entstehen, ein wenig Tongenerator zu spielen und vielleicht den Frequenzgang Ihres Cassettenrekorders oder Tonbandgerätes zu testen, so

probieren Sie bitte das folgende kurze Programm:

```
10 MODE 1
20 PERIODE=10
30 E$=INKEY$:IF E$<>"" THEN PERIODE=PERI
ODE+10
40 LOCATE 1,1:PRINT"PERIODE: ";PERIODE;"
FREQUENZ CA. ";INT(125000/PERIODE)
50 SOUND 1,PERIODE
60 GOTO 30
```

Die Ton-Dauer

legen Sie mit dem dritten Parameter in der Sound-Anweisung fest.
Leider ist es nicht möglich, die Länge einer Note direkt in Vierteln oder Achteln einzugeben, da diese auch immer vom Takt beeinflußt wird.
Aus diesem Grunde haben die Hersteller des CPC vorgesehen, die Notenlängen eines Stückes in 1/100-stel Sekunden anzugeben.

Sollten Sie Musikstücke umschreiben wollen, so setzen Sie zunächst Werte von 25, 50 und 75 für die Dauer einer Note ein, hören Sie sich dann das Spiel an, und korrigieren Sie nach Gehör.

Allein mit diesem Wissen können Sie nun schon eine Reihe von Experimenten und Anwendungen auf Ihrem Schneider CPC realisieren.

Da wären zum Beispiel Musikstücke, die Sie programmieren könnten, oder aber, Sie machen sich daran eine kleine Orgel zu konstruieren.

Übrigens, wenn Sie das Listing von Londonderry-Air eingegeben haben, sollten Sie mit der Sound-Anweisung ein wenig experimentieren, und andere Parameter einsetzen.

So klingt es gleich bedeutend voller, wenn Sie statt des einen alle drei Kanäle arbeiten lassen.

```
10 REM Londonderry-Air
20 READ a,b,c
30 IF b=-1 THEN RESTORE:GOTO 20
40 SOUND 1,a,b,c
50 GOTO 20
60 DATA 338,50,5,319,50,5,284,50,5,253,1
00,5,284,50,5,253,50,5,190,50,5
70 DATA 213,50,5,253,50,5,284,50,5,319,5
0,5,379,75,5,379,50,5,319,50,5
80 DATA 253,50,5,239,50,5,213,100,5,190,
50,5,213,50,5,253,50,5,319,50,5
90 DATA 253,50,5,284,150,5,284,50,5,338,
50,5,319,50,5,284,50,5,253,100,5
100 DATA 284,50,5,253,50,5,190,50,5,213,
50,5,253,50,5,284,50,5,319,50,5
110 DATA 379,75,5,379,50,5,338,50,5,319,
50,5,284,50,5,253,100,5,239,50,5
120 DATA 253,50,5,284,50,5,319,50,5,284,
50,5,319,150,5,319,50,5,213,50,5
130 DATA 190,50,5,169,50,5,159,100,5,169
,50,5,169,50,5,190,50,5,213,50,5
140 DATA 190,50,5,213,50,5,253,50,5,319,
75,5,319,50,5,213,50,5,190,50,5
150 DATA 169,50,5,159,100,5,169,50,5,169
,50,5,190,50,5,213,50,5,253,50,5
160 DATA 284,150,5,284,50,5,213,50,5,213
,50,5,213,50,5,127,100,7,142,50,7
170 DATA 142,50,7,159,50,7,190,50,7,159,
50,7,213,50,7,253,50,7,319,75,7
180 DATA 319,50,7,338,50,7,319,50,7,284,
50,7,253,50,7,190,50,7,213,50,7
190 DATA 253,50,7,284,50,7,319,50,7,379,
50,7,338,50,7,319,175,7
200 DATA 0,-1,0
```

Eine Miniorgel

Zunächst wird Ihre neue Orgel ein Manual benötigen, wozu sich in erster Linie wohl die Tastatur anbietet.
So könnten Sie sich entscheiden, innerhalb der zweiten Reihe die Tasten von S bis L mit den Noten einer Oktave zu belegen.

Aus der Tabelle im Anhang des Handbuches ergibt sich dann folgende Zuordnung:

s	C	478
d	D	451
f	E	379
g	F	358
h	G	319
j	A	284
k	B(H)	253
l	C	239

Die Orgel sieht dann entsprechend aus:

```
100 e$=INKEY$
110 IF e$="s" THEN periode=478:GOTO 220
120 IF e$="d" THEN periode=426:GOTO 220
130 IF e$="f" THEN periode=379:GOTO 220
140 IF e$="g" THEN periode=358:GOTO 220
150 IF e$="h" THEN periode=319:GOTO 220
160 IF e$="j" THEN periode=284:GOTO 220
170 IF e$="k" THEN periode=253:GOTO 220
180 IF e$="l" THEN periode=239:GOTO 220
210 GOTO 100
220 SOUND 1,periode:GOTO 100
```

Eine Oktave steht Ihrem Spiel zur Verfügung, und nach einer kleine Änderung können Sie sogar die Oktaven wechseln.

Dazu bedienen wir uns der Taste -, um eine Oktave tiefer, und der Taste rechts daneben, der mit dem aufrechten Pfeil, um eine Oktave höher zu spielen.

Denn mit jeder Oktave verdoppelt, beziehungsweise halbiert sich die Periode:

```
80 o=1
190 IF e$="-" THEN o=o*2
200 IF e$="^" THEN o=o/2
220 SOUND 1,periode*o:GOTO 100
```

Automatische Begleitung

Wenn wir unsere Orgel noch weiter ausbauen wollen, dann wird in engen Grenzen dank des hervorragenden Basics sogar so etwas wie eine Begleitautomatik möglich.

Denn durch die Anwendung eines Interrupts kann der Rechner gezwungen werden, alle so und soviele 50-stel-Sekunden ein Unterprogramm anzuspringen. In diesem kann natürlich eine Soundanweisung für einen anderen Kanal stehen, so daß dieser die Begleitung spielt, und unser Kanal für das Spiel per Tastatur frei bleibt.

Ergänzen Sie beispielsweise:

```
90 EVERY 30 GOSUB 230
230 SOUND 4,426:SOUND 4,284:RETURN
```

Da für jeden Kanal Tonwarteschlangen, sogenannte Queues, existieren, die jederzeit bis zu fünf noch zu spielende Noten enthalten können, kann die Begleitung sogar einfache Melodien spielen.
Denn das Auffüllen der Warteschlange geht schnell genug vonstatten, so daß sich durch den Interrupt keine Auswirkungen auf unser Spiel ergeben.
Auch hier sollten Sie wieder experimentieren, schließlich steht Ihnen noch ein weiterer Kanal zur Verfügung.

Mehrstimmig Spielen

funktioniert fast genauso.
Stoppen Sie das Programm, löschen Sie Zeile 90, und ändern Sie 220 wie auch 230:

```
220 SOUND 1,periode*o
230 SOUND 4,periode/o:GOTO 100
```

Jeder gespielte Ton wird nun in zwei verschiedenen Tonlagen erklingen.
Doch ist es bei dieser Art der mehrstimmigen Spielweise nicht gewährleistet, das der Einsatz der verschiedenen Kanäle gleichzeitig erfolgt.
Zwar wird es bei diesem Beispiel kaum auffallen, denn erstens wird jeweils ein Ton zu einem Kanal geschickt, und das zweitens auch noch nacheinander in wechselnder Folge, dennoch sollte der Synchronverlauf sichergestellt, und nicht dem Zufall überlassen werden.

Rendezvous

Und so scheint nun auch der Zeitpunkt gekommen zu sein, an dem wir auf die schon angesprochene Rendezvous-Technik zurückgreifen.

Es ist Ihnen bereits bekannt, daß Sie, je nachdem mit welchem Kanal Sie ein Rendezvous ausführen wollen, entweder 8, 16 oder 32 zu dem Code des angesprochenen Kanal addieren müssen.

Was aus dem Schneider-Handbuch jedoch nicht deutlich genug hervorgeht, ist, das **beiden Kanälen die Anweisung zur Ausführung des Rendezvous gegeben werden muß.**

Es reicht bei weitem nicht aus, vielleich SOUND 4+8,248 einzugeben, und dann zu hoffen, daß dieser Ton mit dem nächsten Ton von Kanal A gleichzeitig erklingt.

Vielmehr müßte die Befehlsfolge lauten:

```
SOUND 1+32,periode
SOUND 4+8,248
```

Kanal C muß also wissen, daß er sich mit A arrangieren soll, umgekehrt aber genauso.

So wäre es dann auch schwierig, bei Liedern, die in der Form wie Londonderry programmiert wurden, die Kanäle zu wechseln, oder stellenweise einen weiteren Kanal zur Begleitung dazuzuschalten.
Gedacht ist diese Technik für den Fall, daß die Tonwarteschlangen gleich in einem Schub nacheinander gefüllt werden.

Solch eine Vorgehensweise ist mit Sicherheit die effektivste, und führt zu einem Maximum an Musikausgabe bei einem Minimum von Programmier- und Zeitaufwand. Doch müssen diese Tongruppen, die jeweils zu einem Kanal geschickt werden, dann Synchronpunkte enthalten.

Probieren Sie einmal aus, wie es sich anhört, wenn sie jedem Kanal fünf Noten schicken, dabei aber keine zwei Noten von gleicher Spieldauer sind.
Ohne Rendezvous wird jeder Kanal den nächsten Ton spielen, wenn die Zeit des ersten abgelaufen ist, mit entsprechend gesetzten Rendezvouspunkten wird aber ein Kanal auf den anderen warten.

Den zeitlichen Ablauf macht noch einmal folgendes Beispiel deutlich:

KANAL A	KANAL B	KANAL C
- - B - -	- C A - -	- B - - -

Mit dem Spiel der ersten Note beginnt jeder Kanal für sich allein. Der Beginn der zweiten Note ist für B und C gemeinsam festgelegt.
Die dritte Note in seiner Warteschlange spielen kann C sogleich wenn er mit der zweiten fertig ist, A und B müssen aufeinander warten.

Im Programm werden sich somit folgende Sequenzen finden lassen:

SOUND 1,...	SOUND 2,...	SOUND 4,...
SOUND 1,...	SOUND 34,...	SOUND 20,...
SOUND 17,...	SOUND 10,...	SOUND 4,...
SOUND 1,...	SOUND 2,...	SOUND 4,...

SOUND 1,...   SOUND 2,...   SOUND 4,...

Die Lautstärke

Um die Behandlung des Sound-Befehles nun endlich zu Ende zu bringen, mit dem vierten Parameter regeln Sie die Lautstärke von 'nichts zu hören', 0, bis 'laut',7.
Es sei denn sie haben mit dem nächsten Wert eine von 16 Lautstärke-Hüllkurven festgelegt, dann kann die maximale Lautstärke auch mit 15 angegeben werden.

Die Lautstärkenhüllkurve

dient dazu, den Klang des Tones so zu verändern, das er einem Instrument zugeordnet werden kann.

Denn stellen Sie sich vor, Sie blasen eine Trompete. Dann wird der Ton nicht sogleich beim öffnen eines Ventiles seine volle Lautstärke haben, sondern abhängig vom steigenden Luftdruck wird sie anschwellen, nach kurzer Zeit eine gewisse Höhe haben, und danach wieder abfallen, entweder rapide, wenn Sie das Ventil schließen, oder langsam, wenn Ihnen die Luft ausgeht.

Genau betrachtet sind es sogar vier Stadien, die sich im Verlauf einer Lautstärkekurve beobachten lassen.
Die angloamerikanischen Fachbezeichnungen lauten Attack, Decay, Sustain und Release, wir Deutsche sagen wohl eher Anstieg, Abfall, Aushalten und Freigabe.

Die Werte für einen jeden Bereich lassen sich mittels des ENV-Kommandos angeben, doch wie kommt man an sie heran ?

Am besten durch etwas nachdenken und viel probieren.

So wird ein Schuß vermutlich recht schnell seine volle Lautstärke entwickeln, dann wird die Lautstärke fallen und etwas nachklingen.
Der Lautstärkeverlauf der Pauke eines Schlagzeugs wird sich vermutlich als eine ein spitzes Dreieck darstellende Lautstärkenkurve präsentieren, denn der Schlegel wird das gespannte Fell kräftig durchdrücken, es danach aber auch am Schwingen hindern.

Die Tonhüllkurve

ist in Syntax und Anwendung der Lautstärkenhüllkurve recht ähnlich.
Sie verändert in geringem Rahmen die Frequenz des Tones, so daß ein vibratoähnlicher Effekt entsteht.

Auch hier gilt, daß probieren über's studieren geht, und daß keine festen Regeln angegeben werden können.

STICHWORTVERZEICHNIS

S

T

V

W

X

Z


TEXTOMAT
DATA BECKER



Deutschlands meistverkaufte Datei-verwaltung jetzt in einer speziellen Version für den CPC 464. Erweitert um 80-Zeichen-Darstellung und größere Datensätze mit bis zu 512 Zeichen. Komplett in Maschinensprache und damit superschnell. Läßt sich ideal mit TEXTOMAT kombinieren. **DATAMAT für den CPC 464 kostet einschließlich umfangreichem Handbuch DM 148,–*.**

* Unverbindliche Preisempfehlung



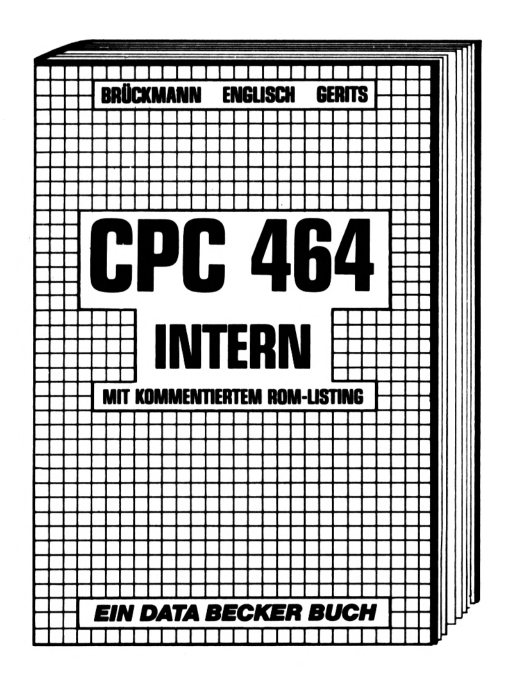
Universelle Buchführung sowohl für private Zwecke als auch zur Planung, Überwachung und Abwicklung von Budgets jeglicher Art. **Komplett mit ausführlichem Handbuch ab April für DM 148,–*.**
In Vorbereitung: **MATHEMAT** das leistungsstarke Mathematikprogramm. Ab Ende April.

* Unverbindliche Preisempfehlung



Das Maschinensprachebuch zum CPC 464 für jeden, dem das BASIC nicht mehr ausreicht. Von den Grundlagen der Maschinenspracheprogrammierung über die Arbeitsweise des Z 80-Prozessors und der Beschreibung seiner Befehle bis zur Benutzung von Systemroutinen ist alles ausführlich erklärt. Dazu Assembler, Disassembler und Monitor als Anwenderprogramme. Einstieg in Maschinensprache leicht gemacht!

Das Maschinensprachebuch zum CPC 464, über 300 Seiten, DM 39,–.



Unentbehrlich für den fortgeschrittenen Basic-Programmierer und ein absolutes Muß für den professionellen Assembler-Programmierer. Z 80-Prozessor, Videocontroller, Schnittstellen sind ausführlich beschrieben. Kommentiertes Listing des BASIC-Interpreters und des Betriebssystems.
CPC 464 INTERN, 1985,
ca. 500 S., DM 69,—.

DAS STEHT DRIN:

Der Schneider CPC besitzt außergewöhnliche Grafik- und Soundmöglichkeiten. In diesem Buch wird gezeigt wie man sie nutzt. Mit vielen interessanten Beispielen und nützlichen Hilfsprogrammen.

Aus dem Inhalt:

- Grundlagen der Grafikprogrammierung
- Zeichensatzeditor
- Sprites, Shapes und Strings
- Mehrfarbige Darstellungen
- Koordinatentransformation
- Verschiebungen, Drehungen, Rotation
- 3-D-Funktionenplotter
- CAD
- Der Synthesizer
- Miniorgel
- Die Hüllkurven

und vieles mehr

UND GESCHRIEBEN HAT DIESES BUCH:

Jörg Walkowiak ist Informatik-Student und erfolgreicher Fachbuchautor (Adventurebücher).

ISBN 3-89011-050-9

Walkowiak / CPC 464 Graphik & Sound

CPC 464


AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
ACME
https://acpc.me/